UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

RECOMMENDATION SYSTEM FOR MODERN TELEVISION

Diogo dos Reis Gonçalves

TRABALHO DE PROJETO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

Trabalho de Projeto orientado pelo Prof. Doutor Francisco José Moreira Couto
e co-orientado pelo Prof. Doutor Paulo Ricardo Pacheco Rodrigues Trezentos

2015

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

RECOMMENDATION SYSTEM FOR MODERN TELEVISION

Diogo dos Reis Gonçalves

TRABALHO DE PROJETO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

Dissertação orientada pelo Prof. Doutor Francisco José Moreira Couto
e co-orientado pelo Prof. Doutor Paulo Ricardo Pacheco Rodrigues Trezentos

2015

Acknowledgements

These nine months wouldn't be possible without the support of most people in my life. I would like to thank my parents and friends for supporting me and for understanding the reason why I didn't have time for them. I thank my University colleagues that helped me reach this academic level and every time I needed inspiration for this thesis.

I would like to thank my supervisors for letting me explore this subject and for guiding me through it when necessary.

I would like to thank my colleagues at Caixa Mágica for the opportunity I had to develop my thesis in this interesting subject and all the support given during its development.

Thanks to every family that was excited to share their television consumption habits with me, allowing me to do a whole evaluation step that otherwise wouldn't be possible.

Finally, I would like to thank everyone who reads my thesis.

Resumo

Atualmente cada vez mais pessoas consomem serviços de televisão e de streaming de vídeo, tanto em casa como na rua usando os seus dispositivos móveis. Um fornecedor de televisão disponibiliza aos seus clientes milhares de conteúdos cujo acesso está limitado no tempo. Assim torna-se necessário que o utilizador decida que conteúdos visualizar. Vários estudos indicam que essa decisão é morosa e que os utilizadores têm dificuldade em descobrir algo diferente do que estão habituados nos canais que geralmente veem. Os métodos habituais para descoberta de novos conteúdos não são suficientes.

Os sistemas de recomendação permitem aos utilizadores encontrar conteúdos que sejam do seu interesse, baseando-se nos seus gostos ou nos interesses de determinada população. Quando os utilizadores entram em contacto com este tipo de sistemas, o seu interesse e satisfação com o serviço aumenta, resolvendo-se assim alguns problemas de procura de conteúdos. Os atuais fornecedores de televisão apresentam sistemas de recomendação básicos que não exploram funcionalidades como a explicação das recomendações e a utilização dos programas vistos pelo utilizador na geração de recomendações. Estas funcionalidades podem ser encontradas em sistemas de recomendação existentes em outras áreas.

O objetivo deste trabalho foi desenvolver um sistema de recomendações para fornecedores de televisão que permite aos seus clientes ultrapassar as atuais dificuldades. Foi desenvolvido um sistema de recomendações que utiliza os dados normalmente existentes num serviço de televisão, apresentando o resultado numa interface apelativa e adequada para o efeito. As recomendações incluem explicações como tentativa de melhorar o interesse do utilizador pelo sistema.

O sistema de recomendações desenvolvido é composto por vários componentes independentes que se ligam entre si e que podem ser integrados num sistema existente. O primeiro componente é o conjunto de fontes de dados que fornecem o sistema. Estes dados incluem informação sobre os programas, os utilizadores e as classificações implícitas e explícitas que os mesmos dão aos programas. De seguida, existe um componente que processa as fontes de dados e converte-os num formato unificado usado pelos restantes componentes do sistema. Durante este processo, é realizada a conversão de classificações implícitas para explícitas, permitindo reaproveitar os algoritmos de recomendação existentes para este tipo de classificação. Esta conversão é realizada através de uma fórmula

desenvolvida nesta tese que tem em conta a relação que existe entre os programas e os episódios vistos pelos utilizadores.

O componente seguinte trata as recomendações. Os dados e as classificações existentes são processados e são fornecidos a vários tipos de algoritmos de recomendação que geram uma lista de programas recomendados. Os algoritmos usados neste processo baseiam-se em técnicas de recomendação existentes como filtragem colaborativa e baseadas em conteúdo. Por outro lado, foram projetados e testados algoritmos criados com o domínio televisivo em mente usando informação sobre o horário e canal de transmissão de cada programa e a relação com os programas visualizados anteriormente. Estes algoritmos foram agrupados através de técnicas de recomendação hibridas, em que se junta o melhor de cada um dos algoritmos para melhorar o sistema, após uma avaliação dos mesmos.

Após a geração das recomendações, existe um componente com o objetivo de adicionar explicações a cada um dos programas recomendados e de agrupa-los por temas relacionados. Neste processo é criado um perfil do utilizador com base nas recomendações, e são gerados grupos de programas com base em propriedades em comum entre eles, como a sua popularidade, a sua categoria, atores em comum, entre outra informação estruturada que seja disponibilizada na fonte de dados original. Estes grupos são ordenados consoante as preferências do utilizador, através do perfil do utilizador gerado. Cada programa recomendado é associado a uma dessas listas. As explicações individuais consistem na relação dos programas recomendados com esse perfil e com outros programas visualizados anteriormente. O resultado deste componente é um conjunto de listas de programas recomendados ao utilizador em que cada programa tem uma explicação associada.

O componente seguinte é responsável por executar o sistema de recomendações sempre que é atualizada a lista de programas disponíveis ao utilizador (tipicamente uma vez por dia nos operadores de televisão existentes) e sempre que existe uma nova visualização por parte do utilizador. O resultado das recomendações é guardado em memória estando sempre pronto para ser mostrado ao utilizador. Isto permite que o utilizador tenha sempre acesso às recomendações e que as mesmas sejam atualizadas sempre que exista necessidade. Esta funcionalidade está disponível através do componente seguinte, os Web Services. O objetivo deste componente é disponibilizar as funções descritas de forma a que possam ser usadas pela interface desenvolvida.

O último componente é a interface de utilizador. Esta interface gráfica apresenta, de uma forma simples e semelhante às interfaces de televisão existentes, as recomendações e as explicações das mesmas ao utilizador. As recomendações estão divididas pelos grupos gerados pelo componente das explicações. Este componente foi usado para testar o sistema de recomendações com utilizadores.

Este sistema de recomendações foi avaliado com duas metodologias de avaliação. A primeira foi a avaliação offline. Esta avaliação consiste em testar os vários algoritmos

criados e usados nesta tese, usando diferentes métricas de avaliação. As métricas usadas foram a cobertura dos algoritmos, o tempo de execução, o erro médio quadrático global e o erro médio quadrático de cobertura. Para este processo de avaliação foi usado um conjunto de dados composto por dados de visualização de programas e por meta dados que descrevem os mesmos. Este conjunto de dados foi dividido em várias partes iguais temporalmente de forma a testar a evolução do sistema ao longo do tempo. A avaliação demonstrou que os melhores resultados globais foram apresentados pelos algoritmos híbridos. O melhor resultado de cobertura foi um dos algoritmos criados especificamente para o domínio televisivo, que apresenta uma diferença significativa quando comparado com os outros algoritmos. No geral, os algoritmos de recomendação apresentaram valores semelhantes para as métricas avaliadas.

A outra etapa das avaliações tratou os testes com utilizadores. Estes testes têm o objetivo de encontrar as preferências dos utilizadores face aos sistemas de recomendação televisivos em geral, validar as recomendações e explicações geradas pelo sistema de recomendações e determinar o interesse dos utilizadores pelos tipos de explicações contempladas no sistema. Para esta etapa foi utilizado o algoritmo hibrido que apresentou o melhor resultado na etapa anterior. A realização destes testes implicou o recrutamento de utilizadores que autorizassem a recolha do seu perfil de visualização televisivo durante um período alargado de tempo, de forma a ter dados para executar o sistema de recomendações. Durante os testes, o utilizador interagiu com a interface do sistema de recomendações e respondeu a perguntas que lhe foram colocadas. Nestes testes concluiu-se que no geral o utilizador era favorável às recomendações e às explicações do mesmo, sendo que este aumentava a sua satisfação e eficiência no processo de escolha de um programa para ver.

Neste momento a área das recomendações está a ter um grande destaque por parte dos operadores de televisão. Neste trabalho propõe-se uma abordagem que apresenta bons resultados por parte da satisfação dos utilizadores e que apresenta técnicas de explicação inexistentes nos sistemas dos operadores existentes. No futuro poder-se-á melhorar este trabalho com a inclusão de novas técnicas de recomendação, outras métricas de avaliação e a realização de testes integrados num operador real em que se possa avaliar durante um período mais alargado a influência das recomendações nos utilizadores.

Palavras-chave: Sistemas de recomendação, Televisão, Explicações, Interfaces de Utilizador

Abstract

Nowadays, people spend more time watching television contents than ever before. Television providers offer their customers thousands of programmes, available for watching for a limited amount of time with recent technologies such as time-shifting, and new ways to watch them such as streaming using their own personal devices. It is impossible for someone to be aware of and to watch every available video, so one has to make the decision on what to watch. Having such a large amount of content available, viewers need to decide on what to watch. Recent research shows that some people get frustrated when searching for something to watch due to the vast amount of programmes which they are presented with.

Recommendation systems enable users to find content which they are interested in through their own preferences while also taking the interests of other users into consideration, allowing their customers to overcome their current difficulties.

The goal of this thesis was to develop a modular recommendation system for television providers. This system uses the information generally available in these environments to predict the user's interest on programmes, including his or her implicit intent. Existing recommendation techniques were studied and evaluated and new TV domain specific algorithms were developed. Techniques were combined to form a hybrid algorithm capable of generating predictions for any programme in the system.

An explanations module was built to generate descriptions that define the reason why a specific item or group of items was recommended. It allows users to be more interested in the recommendations given by the system. Predictions and explanations are then displayed on a user interface.

Two evaluation methods were used to validate the produced recommendation system. Offline evaluations were used to compare the algorithms and it was demonstrated that the use of hybrid algorithms improved the system. Controlled evaluations made by users were also an object of study, through which the system was praised for its general quality, as well as for the availability of the previously mentioned explanations. Most users felt there was an improvement in comparison to existing recommendation systems.

Keywords: Recommender Systems, Television, Explanations, User Interfaces

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Today, more than ever, people subscribe to cable TV providers and streaming services [16, 13]. They spend more time watching television contents than before. They watch television not only at home, in their living room, but also everywhere else using their mobile devices. On a typical cable TV provider, costumers have access not only to live TV but also to VOD (video on demand) services. More recently, cable TV providers started offering week-long time-shifting services for several TV channels, allowing customers to watch their favourite shows at a later time. This means costumers are free to choose between 29,000 different video programmes on a given time [31]. It is impossible for a person to watch every video available so they have to choose what to watch. People also have preferences on what they like to watch. With hundreds of different channels and thousands of time-shifted programmes, people might feel lost when assigned the task of finding something interesting to watch. Studies show that nearly half the people spend more than 10 minutes channel surfing or browsing the EPG (Electronic Program Guide) trying to find something to watch and almost 90% feel they watch the same channels over and over again [16]. More than half get frustrated when trying to find something to watch. Nowadays, the typical ways people have available for finding what to watch on a set-top-box like the EPG are not good enough.

Recommendation systems help users finding content they are interested in and can be based on multiple factors like what they previously watched or what they like. Although missing or hidden on most cable TV providers, clients that have access to recommendation systems find the service helpful, tend to be more satisfied with their cable TV provider and also tend to purchase more content [16, 5].

Most current recommendation systems for cable TV providers deliver limited functionality [1], missing features such as unobtrusively capturing user feedback [34] and recommendation explanations, which are shown to increase user satisfaction and usage [42].

1.2 Objectives

This thesis aims at building an application for set-top boxes or interactive TVs that allows users to quickly and intuitively get recommended TV programmes and movies to watch without requiring any user interaction. It also aims to provide explanations for every recommended programme. Being available directly on the set-top box means users do not need to reach for other sources of information. They can make a choice in the same environment they use to watch TV. To achieve this goal, the work described in the following sections must be studied, developed and evaluated.

Recommendation Engine

The first aim for this thesis is to build a recommendation engine suited for the cable TV reality. Each item, be it a live programme or a time-shifted one, has metadata attached to it. The system should be able to use information from those items and provide recommendations based on that information. Active user interaction with the system like item rating and passive interactions like content watching should also be used as an input to the system. The result will provide the user with a way to find new programmes to watch and new episodes from previously watched programmes.

Explaining Recommendations

Recommendation explanations improve the user's trust and satisfaction in the system [35, 43]. They allow the user to better understand the reason why an item is recommended and to make a decision based on additional information associated with an item. A module to generate explanations will be made to supplement the recommendation engine. Evaluations will be made to find out if the module improves the user's experience.

User Interface

It is important to have good algorithms for item recommendation but the way they are shown to the user have an equal impact on the effectiveness of the system. An interface made to display recommendations in a television environment will be developed and evaluated.

1.3 Contributions

The contributions of this work are:

- A Recommendation System pipeline that can be integrated with a cable TV provider data sources.

- An implicit feedback mapping algorithm that converts implicit user watching sessions to an explicit rating value, allowing existing explicit rating algorithms to be reused.
- Three recommendation techniques built for the TV domain:
  - Item Episode Mean Rating predicts the rating for future programme episodes based on the user's watching history for past episodes.
  - Item Broadcast Time generates predictions based on the time periods when the user watches TV.
  - Item Broadcast Channel generates predictions based on the channels the user watches.
- Two hybrid techniques that combine existing algorithms:
  - Mixed Recommender predicts the ratings by using the best algorithm that covers each item. The chosen algorithms were those that performed better in the offline evaluation.
  - TV Hybrid Recommender uses a set of heuristics that employ different hybridization methods that depend on some known properties about the users and the items.
- An explanations design and algorithm for describing a list of recommendations.
- Offline evaluations with an implicit dataset were performed. When comparing all algorithms, it was shown the TV Hybrid Recommender presented the best global RSME value and Item Episode Mean Rating presented the best coverage RMSE.
- User Studies were performed with users who provided their TV watching history. Most users rated the system efficiency and their own satisfaction positively and wished to continue using this recommendation system in the future.

1.4 Planning

Table 1.1 presents the original milestones and tasks defined for this project. The plan was successfully followed on time with some adjustments. Originally both the recommendation engine and explanations module were under a unique milestone. A new milestone was conceived to allow to better develop a standalone explanations module. Because of this, the interface design milestone was slightly reduced. Finally, the System Evaluation milestone required a preparation phase not contemplated in the original project plan, but that did not affect the defined schedule.

Start date	End date	Duration	Task
09/2014	**09/2014**	**2 weeks**	**Define project goals and requirements**
09/2014	**10/2014**	**4 weeks**	**Research related work on recommendation engines** - Recommendation Algorithms - Recommendation Frameworks - User Interfaces - Existing Video/TV providers recommendation services
10/2014	**11/2014**	**3 weeks**	**Research existing datasets** - Build a dataset
11/2014	**11/2014**	**2 weeks**	**Write the preliminary report**
12/2014	**03/2015**	**15 weeks**	**Develop a recommendation engine adapted for the cable tv reality** - Choose the appropriate technology and recommendation methods - Specification and detailed design - Codification of the system - Tests
03/2015	**05/2015**	**9 weeks**	**Design and build an interface to display recommendations** - Specification and detailed design - Codification of the system - Tests
05/2015	**05/2015**	**3 weeks**	**System Evaluation**
05/2015	**06/2015**	**4 weeks**	**Write thesis**

Table 1.1: Project Milestones and Tasks

1.5 Document Structure

This document is structured as follows:

- Chapter 2 - Related Work - Presents the concepts necessary to understand recommendation systems, explanations, recommendation system evaluation and other related concepts addressed in this work.
- Chapter 3 - Recommendation System - Describes the developed recommendation system pipeline and implementation.
- Chapter 4 - Offline Evaluation - Presents the evaluation steps and results for offline evaluation
- Chapter 5 - User Studies - Presents the evaluation steps and results for user studies
- Chapter 6 - Conclusion - Describes the conclusion and future work.

Chapter 2

Related Work

2.1 Recommendation Systems Approaches

A recommendation system is a tool that assists users by providing suggestions. The user can use that information to make a decision like what products to buy, what books to read or what videos to watch [38, 37]. Generally, a recommendation system suggests items to users. To be able to provide suggestions, the recommendation system needs to predict which items are useful for the user. There are two main types of techniques [2]:

Collaborative filtering is the prediction of what a user likes based on their similarity to other users. A prediction is based on the interactions other similar users had with the system. An item might be recommended to a user as a result of being popular within similar users.

Content-based filtering is based on the analysis of the likeness between the items the user interacts with. The prediction of an item the user might like is related to the similarity with other items the user interacted with.

In addition to these types, some authors [38] also identify other types:

Demographic recommenders provide recommendations based on the demographic profile of the user. It's inspired on the fact that users that share the same demographics might be interested in the same items.

Knowledge-based is a type of recommender that uses a domain knowledge to relate the user requirements with the items properties. This type of recommender is applied in contexts where users typically do not purchase the same items, such as when buying an apartment.

Community-based recommenders make predictions based on data from the friends of the user's friends. They are inspired on collaborative filtering recommenders but they restrict the users that influence the recommendations.

All the mentioned techniques are typically combined to improve the recommendations and to reduce the disadvantages of each individual technique. The resulting method is called the hybrid recommender system. Two or more recommendation methods can be combined to create a hybrid system.

2.1.1 Recommendation System Model

In a typical recommendation system, there are users and items. Let U be the list of participating users and I the list of items the users can interact with. Users can rate items. These ratings are represented in a user-item matrix R. In Figure 2.1, a numeric user-item rating matrix is presented, representing a system where the user would have to rate items with a numeric value from 1 to 5.

	I_1	I_2	I_3	I_4	I_5	I_6	...	I_n
U_1		3			5			2
U_2								
U_3	4			2	3			
⋮								
U_n			5					2

Figure 2.1: An example user-item rating matrix.

Each row represents a user and each column represents an item. The values represent the rating given by a user to an item, meaning item 5 was rated 5 by user 1 and rated 3 by user 3. Other users did not rate item 5. This matrix is generally sparse, most values are missing. Generally there are users with no ratings associated, meaning they didn't rate any item, and also items with no user ratings. The objective of the recommendation system is to generate a list of items it predicts the user would be interested in. In other words, a recommendation system should be able to fill the missing values of the matrix. In this example, to predict which value would user n rate the item 5, the system would compare the user current ratings with the ratings of the users that rated item 5.

This is generally known as the top-N recommendation problem [14]. A list of recommendations is defined as follows:

> Given a user-item matrix R and a set of items I that have been rated by a user, identify an ordered set of items X such that $|X| \leq N$ and $X \cap I = \varnothing$

2.1.2 Collaborative filtering

Recommendation systems based on collaborative filtering methods allow users to make decisions based on the opinion of other users [36].

Collaborative algorithms depend on the feedback not only from the current active user but also by the users of the system. It is necessary to store the feedback provided by the

user. In collaborative algorithms, user ratings are generally represented in a user-item matrix such as the one presented in section 2.1.1.

There are two main types of collaborative filtering algorithms: memory based [15] and model based [7], both detailed in the following sections. But first, baseline techniques are presented and detailed.

2.1.2.1 Baseline Predictors

Baseline techniques provide a way to get non personalized recommendations and to have a base value to compare other algorithms. They can also be used as a fallback when other algorithms of the system are not able to provide a recommendation for the desired user-item pair [19]. The most common techniques use the average user or item ratings as the predicted rating.

Item Mean Baseline
This technique returns the mean rating of an item for all predictions, based on the ratings provided by other users. A common implementation [19] defines it as follows:

$$r_{u,i} = \mu + b_i$$

$r_{u,i}$ is the value of the predicted user-item rating. μ is the global mean rating for all the items in the system, and b_i is the item's average rating, without the global rating. It is calculated with the following formula:

$$b_i = \frac{\sum_{k=1}^{n} x_k - n\mu}{n + \gamma}$$

The number of existing ratings for item i is n. x_k is the value of one of those ratings. γ is an optional damping factor.

User Mean Baseline
A technique similar to the previous one which instead returns the mean user rating for all predictions.

$$r_{u,i} = \mu + b_u$$

Where b_u is the user's average rating, defined as

$$b_u = \frac{\sum_{k=1}^{n} x_k - n\mu}{n + \gamma}$$

User and Item Mean Baseline
A generic variation of the previous baseline techniques can be defined as

$$r_{u,i} = \mu + b_i + b_u$$

This allows the baseline prediction to be calculated using both previously defined formulas, returning a rating that is based on the average user ratings and in the average item ratings from the other users.

2.1.2.2 Memory Based

Memory based algorithms use the entire matrix of ratings directly when calculating recommendations. Their first step is to find the user neighbors, a list of similar users, by correlating user's ratings on items. Then, they calculate the user preferences over an item comparing the user's rankings with its neighbors.

An example formula of a memory based implementation for item based filtering:

$$r_{u,i} = \frac{\sum_{j \in K_u(i)} w_{i,j} r_{i,j}}{\sum_{j \in K_u(i)} |w_{i,j}|}$$

$K_u(i)$ is the list of items rated by user u similar to the item i. $w_{i,j}$ is the weight the item has in the calculation. This list can be generated by any similarity computation method presented later in this Section.

Memory based algorithms are typically partitioned in multiple components that have an impact on overall quality of the recommendation system [15]:

Item based and User based filtering

Collaborative filtering can be user based or item based. User based methods are based on the opinion of users with similar tastes when providing a recommendation. Item based methods rely on the ratings given to similar items. Both types are based on similar techniques, described in this Section.

[15] identifies five criteria to help to choose between both types of filtering: accuracy, efficiency, stability, justifiability and serendipity. They depend mainly on the number of users and items of the system and the number of ratings pairs.

Rating Normalization

Different users might have the same opinion on an item but assign different ratings to it. A user might only use the positive values on the rating scale and another user might be more open to use the full scale. Although they share the same opinion on a movie, they rate it differently. This discrepancy when rating the same opinion is problematic for recommendation algorithms. The following rating normalization algorithms may be used to overcome this situation:

- **Gaussian normalization method**

  This approach, also known as mean-centering, was proposed by [36]. It suggests a solution to the shift of average rating by subtracting the user ratings from the

user averages. It also proposes a solution to the different rating scales problem by dividing the ratings with the variance:

$$\hat{r}_{u,i} = \frac{r_{u,i} - \overline{r}_u}{\sqrt{\sum_i (r_{u,i} - \overline{r}_u)^2}}$$

This formula can also be used by item-item collaborative algorithms by replacing the user average ranking with the item's average.

- **Decoupling normalization method**
  This method, suggested by [25] calculates the probability for the item to be preferred by the user. If there is a high probability the user rates items with a value less or equal than a specific rating, then it is reasonable to assume those items are preferred by the user.

  $$Pr(R\ is\ preferred) = Pr(Rating \leq R) - Pr(Rating = R)/2$$

  R is a rating value. The first term calculates the probability an item is rated a value less or equal than R. The second term calculates the probability an item is rated R. A value with the majority of ratings means it is not a special rating, so the second term is subtracted to the first term, reducing the preference over values with many ratings.

- **Baseline Subtracting User Vector**
  A baseline function, like the ones identified in Section 2.1.2.1, can be used to normalize a user vector [19].

  $$\hat{r}_{u,i} = r_{u,i} - b_{u,i}$$

  The normalized rating is calculated by subtracting the baseline value to the original user rating. Its usefulness depends on the chosen baseline function.

Similarity Computation

Similarity functions are used to calculate how similar two items or two users are. A higher value means they are more related to each other. For example, similarity between two users should be high if both users rated the same items equally, meaning they have identical tastes.

- **Cosine-Based Similarity**
  A similarity measure that transforms each user ratings in a vector and calculates the cosine between both vectors

  $$w_{u,v} = cos(\overrightarrow{u}, \overrightarrow{v}) = \frac{\sum_{i \in I} r_{u,i} r_{v,i}}{\sqrt{\sum_{i \in I_u} r_{u,i}^2} \sqrt{\sum_{i \in I_v} r_{v,i}^2}}$$

  Each vector contains the ratings the user gave to the items and zero for unrated items. This method does not take into account the effects of mean and variance.

- **Pearson Correlation Similarity**
  A correlation-based measure, used by [36], that removes the effects of mean and variance.
$$w_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r}_u)^2}\sqrt{\sum_{i \in I}(r_{v,i} - \overline{r}_v)^2}}$$

Neighbourhood Selection
A large recommender system might have millions of items and users to use as an input to the recommendations. It is difficult to store all those values in memory. According to [15], there are a number of techniques to filter the number of neighbours used in the computations to the best candidates.

- **Top-N Filtering** A list of the N nearest neighbours, the most similar, is kept, for each user. The value of N must be carefully selected. [21] found N = 20 is a good starting point with a value up to 50. They found higher values don't bring better results.

- **Threshold filtering** Keep a list of all the neighbours with a similarity weight larger than a chosen threshold. A high value means only high quality neighbours are kept. Unfortunately there might not be sufficient users to keep neighbours for every user. The ideal threshold value depends on the data and must be evaluated. [41] evaluated different thresholds values with multiple algorithms.

- **Negative filtering** Discard negative rating correlation data and only keeps the other values. Negative values generally don't have a significant impact on the results [15] as. it depends on the dataset.

2.1.2.3 Model Based

Instead of using the full matrix when generating recommendations, user ratings on items can be used as training data to build a model that recognizes patterns and makes predictions. Model based algorithms use machine learning and data mining techniques [7] to build such kind of recommender. Predictions are generated based on the previously built model. A common model based algorithm is the SVD factorization technique.

SVD-Based
Singular Value Decomposition (SVD) is a matrix factorization technique. [39] suggested the application of SVD to the collaborative filtering problem. SVD is the factorization of a matrix R in three other matrices:

$$R = U \cdot S \cdot V^T$$

When applied to the collaborative filtering problem, R is the user-item matrix, U and V are the user and the item matrix and S is a diagonal matrix with the rank of matrix R as its side. After doing the factorization this technique allows you to reduce the side size of S and lowering the size of the other matrices U and V^T. This means an approximation of an original matrix R can be stored in much less size. This value is known as the number of features a user and an item are represented. Those features are automatically inferred and denote characteristics that identify the user and the items, such as action movie. Unfortunately, generally it is not possible to identify what the features mean. A prediction is then generated by calculating a dot-product between matrices.

To use SVD it is required that the matrix must be complete, with no missing values. The user-item matrix has to be prepared to work with this technique. [39] suggests to use the averages to fill the matrix missing values. Others suggest alternate algorithms to calculate SVD using only the known values, such as FunkSVD [20].

2.1.2.4 Slope One

Slope One [28] is a fast alternative to the presented memory and model based collaborative filtering schemes. The authors consider it is a fast algorithm and easy to implement. Their tests presented similar results when compared to typical collaborative algorithms.

This method is based on the difference between two items' ratings. Assuming a user u and a set of items $S(u)$ that represents the items rated by u, they can calculate a prediction for item i. First, they calculate the difference between the ratings the other users made between item i and the other items in $S(u)$. Then, they calculate the average of the sum between the rating u made to each item from $S(u)$ and its respective average.

The formula is defined as:

$$P(u)_i = \frac{1}{card(R_i)} \sum_{j \in R_i} (dev_{i,j} + u_j)$$

dev_i, j is the average deviation between the rating of two items i and j, R_i is defined as $R_i = \{j | j \in S(u), j \neq i, card(S_{i,j}(x)) > 0\}$ and $card(R_i)$ is the number of items in R_i.

There are two variations of this scheme that try to overcome some limitations: The Weighted Slope One scheme takes into consideration the number of ratings observed for each rating. If an item has more ratings than another, it should be more trusted when calculating the prediction. This variation adds an associated weight to each item. The other variation, Bi-Polar Slope One, adds the concept of liked and disliked items, and takes only into considerations ratings of the same type when calculating the deviation.

2.1.3 Content-based filtering

Content-based recommendation systems use the item's content and metadata to provide recommendations. The recommendation system analyses the items and provides recommendations based on the associations between them so it is fundamental to be able to represent the item's data in a structured way.

A content-based filter is generally structured in three different components, according to [29]:

Content Analyser

The Content Analyser's purpose is to present the documents in a way suited for processing by the other components. It analyses the unstructured documents and provides them as structured data, e.g. an unstructured text can be converted to a structured keyword list.

Most common content analysers use a Vector Space Model representation and information retrieval techniques such as word stemming and TF-IDF [29].

Other techniques can be used, [32] compares the usage between Vector Space Model, Random Indexing and Logistic Regression for a TV-show recommender. [29] describes semantic analysis using ontologies.

Profile Learner

This component combines the user feedback with the data provided by the Content Analyser to build a user profile. A user profile contains the user preferences in the context of the recommendation system. For example, it can be a list of the TV programme categories with its associated weight.

Filtering Component

This component takes a list of items to recommend and compare them with the user preferences on the user profile. Any algorithm like Cosine Similarity can be used to calculate the similarity between the items and the users.

2.1.4 Hybrid filtering

Hybrid algorithms are used to combine the strengths of multiple recommendation algorithms. There are multiple techniques to combine the algorithms. [8] identifies the following techniques:

Weighted A weighted recommender combines the scores of all algorithms to produce a weighted score. Each algorithm has an associated weight that may be adjusted.

Switching This technique decides which algorithm to use depending on some programmed criteria. For example, a system may use a fallback algorithm if it doesn't have confidence to use the main algorithm.

Mixed A mixed recommender shows recommendations from multiple algorithms at the same time.

Cascade Cascade methods use first a recommender algorithm and then another to refine the recommendation.

Most real world recommendation systems, such as those identified in Section 2.7, use hybrid filtering methods.

2.1.5 User Feedback

Generally, the most common recommendation systems approaches require and assume that the provided user feedback is an explicit intent like an item rating, as shown in the previous sections. Users can explicitly give their opinions on items using a numeric scale ranking [36] or by selecting a binary value such as a "like" or "dislike" [3]. These types of ratings can also appear in other forms, like the user assigning stars to an item instead of assigning a number.

Explicit feedback is not the only approach the system has to get the user interests. When interacting with a system, users also implicitly provide their opinions. This information can be extracted by analysing the user interactions with the system such as purchased items or watched items. This is known as implicit feedback [34]. Collaborative algorithms are generally tweaked to work with explicit feedback, but they can be tweaked to work with implicit feedback by value mapping [5, 24].

2.2 Explaining Recommendations

A recommendation system typically generates a list of one or more items to recommend to a user. The user after looking to the information provided by the system has to decide if those items are worth exploring. If the system just provides a list of items, it doesn't provide any additional information that may help the user decide or known the reasoning behind the recommendations [22]. Recommendations Explanations try to solve this problem by backing the recommendations with explanatory data.

2.2.1 Explanation Aims

Explanations in recommendation systems are built with different purposes in mind, depending on the aims of the system developer. [42] distinguish 7 different aims found in current recommendation systems that identify the purpose of explanations:

Transparency An usability principle [33] explaining how the system works. There are critical systems where it is fundamental the user knows and understands how the it works and what it did to provide a recommendation, like medical systems.

Transparency is often discussed as a component of white box and black box [22], or transparency and justification. The black box model supports that explanations are produced independently of the underlying recommendation algorithm. The white box model supports the usage of explanations to represent how the algorithm processed the items.

Scrutability It allows the users to tell the system it is wrong. The system should provide the necessary means so the user knows the recommendation is wrong and is able to correct it.

Trust Increase users' confidence in the system. Transparency and interaction with the recommendation system increases users' trust. Users come back to use the recommendation system if they trust the system.

Effectiveness Help the user to take good decisions.

Persuasiveness Convince the users to try or buy. A recommendation system might change the will the user has to consume an item. [22] studied 21 different interfaces recommending the same movie and ranked them by persuasiveness.

Efficiency Help users to make decisions faster. It is considered to be another usability principle [33].

Satisfaction Increase user enjoyment and ease of use. A recommendation system may be made to be fun to use. Users may like the addition of a recommendation system.

A recommendation system can't do well in the 7 listed criteria so a trade-off has to be made. For example, [42] consider that satisfaction in recommendation systems for TV shows are more important than effectiveness.

2.2.2 Presenting Recommendations

Recommendations can be presented in multiple ways. How they are presented influences the used explanation methods. [42] identify these forms of explanation presentation:

Top Item The recommendation system presents and explains just the best item for the user.

Top N-items Show multiple related items at once. The explanation may be at the level of the group or at the level of each individual item.

All items Present every recommended item to the user and allow him to navigate through them and find why an item has an high or a low rating.

Similar items Present a list of similar items when one is selected.

2.2.3 Explanation Styles

As we have seen there are many things that affect the objective of an explanation, so there are many things that can be added to a recommendation system that work as an explanation. These explanation styles may be related to the underlying recommendation system, but that is not necessarily true. According to [43] there are the following explanation types:

Collaborative-Based Explanations based on other users' behaviour. These kind of explanations allow the user to find what similar users, or users in the same situation did when interacting with items. In [22], of the 21 different collaborative-based interfaces, the most preferred by the users was a graph that represented the number of similar users that rated the movie positively, neutral or negatively.

Content-Based Explanations based on the content of the recommended items. Although these recommendations can also be based on user similarity, the explanation is based on a property similar between the items rated and recommended items. A movie recommendation system may recommend a movie because the user likes a participating Actor.

Case-Based Reasoning Comparison between recommended items and previously rated items, omitting the similar properties between them.

Knowledge and Utility-Based Explanations based on the similarity between the user needs and the items properties.

Demographic Explanations based on the demographic information about the user.

2.3 Evaluation

Evaluating recommender systems allows to help choosing between multiple candidate algorithms and to analyse if they are improving the user experience. There are multiple evaluation methods available. According to [40], there are three types of evaluation for recommendation systems:

Offline Experiments These kind of experiments typically use previously collected data on the user's behaviour with the system, so they don't require real users. This data can be used to determine if the recommendation system can predict the users'

behaviour. It is of limited use because it assumes the user will behave or make the same choices with the presence of a recommendation system but it is low cost and easy to use. Generally the collected data is partitioned in a training set that the recommendation system uses to generate predictions and a test set, a portion of the data that is used to compare the predictions with the real user ratings.

User Studies Tests with recruited users. We have them interacting with the system and observing what they do, and then making them questions before, during and after the test. This kind of tests can be used to evaluate the influence of the recommendation system and the overall user satisfaction. They give the most granularity of data and feedback but they take much time and are costly, requiring to have a set of users willing to do the tests. Tests with recruited users are effective when studying TV recommendation systems. [1] tested their TV recommender interface in a controlled laboratory emulating a living room environment.

Online Evaluation These tests involve deploying the recommendation system to a running service and evaluating how the users behaviour change. We can compare systems and determine which one the users interact with more. Users are normally unaware they are being used for testing. These kind of tests give the best evidence of the usefulness of the system. They require an existing system with real users and we have to be careful how we choose the users and how the changes are displayed, to be sure they represent the whole user base of the system.

2.3.1 Evaluation Measures

Generally, studies assume that a good recommendation system is one that correctly predicts the items the user prefers [23]. Depending on the objective of the recommendation system there are adequate accuracy measures that may be used.

In recommendation experiments the most common objective is measuring the accuracy of ratings prediction. The most used metrics are based on the Mean Absolute Error (MAE) metric [23]. The successful Netflix contest used the Root Mean Squared Error (RMSE) metric to compare the competing algorithms, and inspired most related research to be based on that metric [4].

- **Mean Absolute Error (MAE)**
  For a recommendation algorithm f and a test set R_{test}, MAE is defined as [15]:

$$MAE(f) = \frac{1}{|R_{test}|} \sum_{r_{ui} \in R_{test}} |f(u,i) - r_{ui}|$$

  A lower MAE means the algorithm produced predictions with less error than a higher value

- **Root Mean Squared Error (RMSE)**
  RMSE compared to MAE penalizes large errors on predictions.

$$RMSE(f) = \frac{1}{|R_{test}|}\sqrt{\sum_{r_{ui} \in R_{test}} (f(u,i) - r_{ui})^2}$$

  Both these metrics are sometimes presented in a variant normalized to the ratings scale.

Multiple TV related studies [24, 5] acknowledge these metrics can be useful but propose alternative metrics they feel are more adequate to implicit TV feedback. An Italian IPTV recommender system [5] was evaluated with a leave-one-out approach. For each user they removed each rated item individually and then checked if it appeared on the newly generated top 5 recommended items. The recall metric was the percentage of hits. To evaluate another recommender system, [24] built a measure named $\overline{rank}$, that compared the actual watched data with their algorithms proposed ranking. Lower values in this metric meant the watched items were better ranked.

Some studies consider that although accuracy metrics are useful they don't capture some aspects of user satisfaction [30, 45]. Other fields like items coverage and serendipity should also be considered.

2.3.2 Datasets

Due to the importance of recommendation systems evaluation, there have been multiple efforts in assembling and providing datasets. Such datasets allow to train and validate algorithms.

The most common datasets used in movie recommendation systems are:

MovieLens Collection of datasets by GroupLens with diverse quantities of explicit ratings collected through MovieLens from 1995 to 2009.

Netflix This dataset was assembled for the Netflix Prize contest for collaborative filtering algorithms [4]. It contains over 100 million explicit ratings made in the Netflix service between 1998 and 2006.

MovieTweetings Dataset based on explicit public ratings made by Twitter Users in IMDB [17]. This dataset contains data collected since 2013 and it is updated daily. It is possible to extend this dataset with information from Twitter because, unlike the previous datasets, the user identities are public.

2.4 User Experience

Users need to have a way to interact with the recommendation system for it to be useful. A recommendation system for set top boxes and smart TVs must have special considerations. On one hand, it should be adapted for TV navigability and be usable with a remote control and a big screen. People don't usually like to waste much time when navigating on TV [16], so on the other hand it needs to display recommendations well so users don't lose themselves or stop using the system.

There are multiple areas to explore when developing a recommendation system user interface [44, 19]:

2.4.1 Soliciting Feedback

Soliciting feedback in TV recommendation systems should not distract the user from watching TV [44]. As explained earlier, users provide feedback while they are interacting with the system. The system unobtrusively gathers implicit feedback from user interactions. Generally recommendation systems ask for explicit feedback. It can be provided by the means of a numeric scale [36] or a discrete scale [3]. When designing its presentation and interaction method, users are influenced by the component's colour, size, scale range and images [44].

2.4.2 Presenting Recommendations

After generating recommendations, the system must communicate them to the user. Typically the algorithms provide a value of how relevant the recommendation is to the user. That value can be used when presenting the recommendation to the user. Generally it is used to order the ranking showing the most relevant first. Other approaches use it to highlight the most relevant recommendations, by showing it with in a larger space [27]. Presenting recommendations is also related to providing explanations as detailed in Section 2.2 and in the next Section.

2.4.3 Providing Explanations

Explanations, detailed in Section 2.2, influence recommendation presentation. When users ask others for recommendations they can ask why they get those recommendations. Explaining recommendations may improve the users trust and satisfaction in the system [35]. The user interface should be able to show the user why such recommendation is given.

2.4.4 Big Screen Interface

Due to the input interface limitations and the typical distance the user has to the TV the interface must respect specific user experience guidelines. There are multiple modern guidelines such as those provided by Google[1], Samsung[2] and Amazon[3]. These guidelines suggest how to provide an intuitive navigation

2.5 TV Recommendation Systems

Over the years there have been studies about the implementation of TV recommendation systems.

- TV Advisor [11] was a project that researched the usage of implicit vs. explicit user profiling and what kind of recommendation algorithm fitted the best to the problem of TV programmes recommendations. The service was available as an HTML interface.

- Personal EPG [18] used user profiles to match them with TV programmes. It introduced a method to create profiles automatically by observing channel tuning and then by clustering algorithms. The authors built a recommender interface adapted for interactive TV terminals.

- PTV, a personalized TV guide [10] was a recommender system, that the user could interact with in a PC and WAP based devices over the Internet. It collected the user preferences by asking him to fill a form on the first run and then by collecting feedback on the programmes. It included both collaborative and content based algorithms, depending on the context.

- TV Scout [6] was a web based filtering system that solved the cold-start problem by presenting itself as a retrieval system and lowering the user barrier to entry by not asking the users preferences upfront.

- TiVo [3] is a set-top-box that provides TV recommendations based on item-item collaborative filtering algorithms. It provides recommendations by showing a list of programs the user might be interested in and by auto-recording recommender programmes.

[1] https://developer.android.com/design/tv/patterns.html

[2] http://www.samsungdforum.com/UxGuide/2014/01_principles_for_designing_applications_for_samsung_smart_tv.html

[3] https://developer.amazon.com/public/solutions/devices/fire-tv/docs/design-and-user-experience-guidelines

- TV Predictor [27], a recommender application for smart TVs and mobile devices using industry standards to content management. It provided recommendations for both implicit and explicit recommendations. It also contains an auto zapping features, meaning the system will change the channels automatically based on its recommendations.

2.6 Frameworks

There are many appropriate recommendation frameworks to use as a base for a recommendation system. They range from a simple algorithms collection to a full system ready to integrate on a current application. This list focus on recently updated and maintained open-source Java tools. In the end some alternative tools are mentioned.

LensKit

LensKit is a framework allowing the construction, testing and evaluation of recommendation systems [19]. Its base implementation focuses on collaborative filtering algorithms and ratings but there are some community extensions to add content-based filtering algorithms. It is designed to be easily extensible. The system receives user feedback and items as input and returns a Top-N recommendation or a predicted rating. This framework also includes tools to test and evaluate the resulting system and to compare algorithms. Overall, the tool was built with research in mind and it is referenced in many research papers.

easyrec

easyrec[4] is a ready to use recommendation system. It integrates with current applications using a Rest API. One of its main advantages is it does not need an expert on programming or recommendation systems to be used. It includes a graphical user interface to setup the algorithms. It also has features for the advanced users allowing it to be extended with new recommendation methods. This framework works with collaborative filtering methods. It also contains plugins to add basic content-based filtering but it requires manual configuration.

LibRec

LibRec[5] is a library for recommendation systems. It contains a collection of state-of-the-art recommendation algorithms. It is heavily focused on algorithms research. This library contains an evaluator for multiple measures for recommendation systems testing.

[4] http://easyrec.org/
[5] http://www.librec.net/

Sifarish

Sifarish[6] is a collection of recommendation algorithms and solutions based on Apache Hadoop.

In addition to these Java frameworks there are outdated alternatives and other languages frameworks. Duine framework is similar to LensKit and has an Explanations API, but its last update was in 2009. PredictionIO is similar to easyrec but less modular. LibRec is often compared to MyMediaLite, but the programming language is C#. There are also more broad tools like Apache Mahout and Weka.

2.7 Real World Cases

In this Section I present some commercial and research projects that use and depend on recommendation systems.

Netflix

Netflix is one of the most popular online video streaming services for on demand movies and TV series. Content recommendation is a fundamental research area for their business model. In 2006 they launched the Netflix Prize competition offering prizes to those who build the best collaborative filtering algorithms. This competition has generated an increase in research and innovation in the area. The winner algorithm used hundreds of predictive models to reach the final goal.

The Netflix service is built around recommendations [4]. The home screen, the first screen the user sees, features dozens of personalized recommendations. This emphasis shows not only it is important to have good recommendation algorithms but also a good user interface design. While developing the recommendations system, Netflix took into account several criteria like optimization for accuracy and diversity, knowing that an account can be shared by various family members and people have different moods when using the service at different times. They found it is important that the user is aware of why the contents are recommended through explanations. Items are ranked by weighing popularity of an item with its predicted rating for the user.

YouTube

YouTube is the largest community driven video sharing website. It allows users to watch and upload their own videos. One of the challenges they face is content discovery and

[6] https://github.com/pranab/sifarish

recommendation [12]. Users have to find new and interesting videos easily. Their recommendation system is a top-N recommender. It is based on the videos metadata and the user's interactions with the content. There is implicit interaction like the user video history and explicit like video ratings. The purpose of the system is to recommend high quality videos relevant to the user interests.

Recommended videos are those the user will most probably watch after having watched an initial video. This definition is the base of the recommendation system. They use association rule mining and co-visitation counts techniques.

User Interfaces are considered very important. Every recommendation contains the videos basic information like the title and the thumbnail. It also contains a basic explanation on why the video is being recommended with a link to the previously watched video that triggered the recommendation.

2.8 Challenges

Below is a list of challenges identified in literature and studies about the subject.

1. **Implicit Feedback** Generally recommendation systems use the user preferences over items to calculate recommendations. For example, a service might ask the user to rate books in order to have data on user preferences. In the context of television, users might be asked to rate TV programmes in order to know their preferences. Unfortunately that would reduce the usefulness of the system because it would only work if the user rated the items and not every user is interested in that cognitive load [34]. It must be identified a way to obtain the user preference without asking for the explicit ranking. An implicit ranking can be calculated using data about the interaction of the user with the system, like how long a user took to watch a program. Such kind of ranking is not perfect, e.g. a user may like a program he didn't have time to finish. The challenge here is how to take this implicit feedback and transform it on something that reflects the user preference.

2. **Sparse Data and Items** The items available to the users change daily. A program is only available for a few days unless it is rebroadcast. So it is difficult to have multiple users to give feedback on the same item. The recommendation system should be able to recommend items even if no user watched them.

3. **Who's watching?** A television set is normally used by different members of a family. One member's preference is different from another. The feedback fed to the recommendation system may refer to the tastes of different users. A recommendation system should be able to identify the user or be prepared to do recommendations for the whole family.

4. **Real-time requirements** If the user asks the system for recommendations, but it takes much time to answer, his mood will drop and he will not use that function again. Also, the items available change daily. An item today might not be available tomorrow. A recommendation system must be prepared to continuously update the items it recommends and be fast as possible to not bore users.

5. **Quality of metadata** To recommend programs based on their content there has to be information available about them. It's not an easy task to classify their content automatically because they are based on video and audio streams and sometimes their content is made live (for live TV programmes). For that reason, the content metadata has to have quality and include information like the program actors, categories, etc.

2.9 Summary

This chapter introduces the necessary concepts to understand this work and used techniques. It starts by describing recommendation systems and its types of techniques. Then the two main techniques, Collaborative filtering and Content-based filtering, were detailed. These techniques are combined in the next chapter to form a recommendation system.

The steps required to design explanations were introduced in this chapter. In the next chapter they are used to design a new explanations module. User experience for recommendation systems was explored in this chapter and is the basis for the interface developed in the following chapter.

The survey of academic and commercial recommendation systems allows to see what they solve and what can be explored in the next chapter, such as the used algorithms. Some of the presented challenges are developed in this thesis: A mapping function is built to contribute to the Implicit Feedback challenge, content-filtering algorithms were used for the Sparse Data and Items challenge and the Real-time requirements challenge was explored with the development of a dispatcher module.

Offline Evaluation and User Studies were two evaluation techniques presented in this chapter and are used to evaluate the developed system.

Chapter 3

Work

3.1 A TV Recommendation System Model

The recommendation system model used in this thesis is based on the model described in section 2.1.1. In a TV recommendation system, U is the list of participating users and I the list of programmes the users can interact with. A television programme contains one or more episodes E. An episode is a broadcast slot assigned for a programme that is not a repetition. This means a movie would contain one episode and a recurring programme like a TV Series would contain many episodes.

Each user expresses their interest for items by rating them. This rating is expressed on a scale of 0 to 10, meaning that a higher rating means a higher preference on watching an item. The relation between the users and the items is represented in a user-item matrix, as explained previously in Section 2.1.2.

Ratings may be given either explicitly by the users or generated by using implicit data from user interaction with the system. In this context, one possible source for implicit data would be the watched time for each item.

The main objective of a recommendation system for set top boxes is to generate lists of recommendations. The recommendation system generates multiple recommendation lists that are used through the set-top-box application. These lists are dynamic and specific to the user context.

3.2 System Modules and Pipeline

To produce a recommendation system, it is necessary to develop multiple components that will be connected together. The developed components and pipeline is presented in Figure 3.1. This pipeline allows the integration in an existing system by replacing just the input and output components. Its data model is based on the TV Recommendation System model presented in the previous section.

1. **Data Sources** The source information, also known as dataset, required to generate

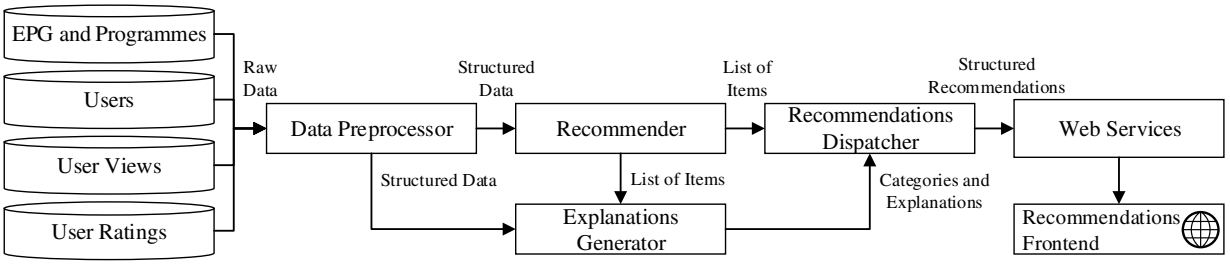

Figure 3.1: Pipeline Overview

recommendations, detailed in Section 3.2.1.

2. **Data Preprocessor** This component converts the information provided from the data sources to the format expected by the recommender. It also processes implicit mapping, as detailed in Section 3.2.2.1.

3. **Recommender** The recommender component uses the algorithms detailed in Section 3.2.3 to analyse and generate lists of recommendations.

4. **Explanations Generator** This component makes a user profile based on the recommended items and organizes the items in personalized lists for the users. It provides an explanation for each recommended item, as described in Section 3.2.4.

5. **Recommendations Dispatcher** This module, presented in Section 3.2.5, is responsible for caching and refreshing user recommendations. It allows the system to provide recommendations even when the recommender component is generating new ones.

6. **Web Services** Allows the frontend to request user recommendations to the recommendations dispatcher component.

7. **Recommendations Frontend** The frontend where the recommendations are displayed, presented in Section 3.2.7

3.2.1 Data Sources

Recommending television programmes requires access to multiple types of data. The data can come from one or multiple different data sources. The following list enumerates the required data types and the information expected from them. It is based on the model presented in Section 3.1.

1. **EPG and Programmes** The list of items to recommend, including programmes which have been or are soon to be broadcast. This kind of data is generally structured as an EPG for live TV and lists for VOD content. These items usually have

associated metadata, like the broadcast date, a small description, its category and participating people.

2. **User Metadata** The list of users that interacted implicitly or explicitly with the system. It may optionally contain the interests and demographics of the users.

3. **User Views** The largest source of feedback from the users is the visualization data for the programmes. It is an implicit source of information because no extra action from the users is necessary besides their channel selections.

4. **User Ratings** The other source of feedback is the ratings the users give to the items they watched. This is an explicit source and so generally it contains less data.

3.2.2 Data Preprocessor

Although the required data sources are all from the TV domain, its raw data is generally structured in a unique way, depending on their format and original source. This component takes the raw data from the data sources and converts it to the format used by the recommender system, presented in Table 3.1. This allows to reuse and integrate this system with any existing infrastructure that provides this information, just by replacing this component. An infrastructure that provided data using multiple CSV files would require a different data preprocessor implementation when compared to a system with its data structured and saved in a database.

Data Type	Parameters
User	User Id, Registration Date, List of Properties, List of Sessions, List of Ratings, List of Programmes
Programme	Programme Id, Title, List of Properties, List of Users, List of Episodes
Episode	Episode Id, Parent Programme, Start Date, Duration, Season Number, Episode Number
Session	Session Id, Episode Id, User Id, Start Date, Duration
Rating	Rating Id, Program Id, User Id, Date, Rating Value
Property	Property Id, Property Type, Property Value

Table 3.1: Structured data after running the preprocessor module

A user watches episodes and rates programmes. A programme has one or more episodes. A property represents a piece of metadata associated to a programme. This metadata can be an actor, a director, a year, a description, or any other metadata available on the original data source.

As part of the data conversion, this component is tasked with mapping between implicit and explicit feedback. This process allows to use algorithms only intended for explicit feedback and is described in the next Section.

3.2.2.1 Implicit Feedback Mapping

Most recommendation algorithms and evaluation measures are based on explicit feedback by the users, as explained earlier in Section 2.1.5. Although it is possible to collect and use explicit feedback on TV programmes, there are advantages in using implicitly collected feedback. When watching TV, users generally tend to minimize interactions. Asking for the user opinion on each watched programme is not necessary when they implicitly disclose that information when they choose to watch a programme. This technique is common in other fields like search engines. Instead of asking the user to rate the web pages they visited, the click data on the results page and time spent on the web pages are commonly used as implicit feedback [26].

In this thesis the goal was to find a way to convert the implicit user intents to an explicit value representing the rating the user would give to an item. As seen in Section 2.1.5, research has shown there is a good correlation between both types of feedback. Research based on explicit ratings algorithms and metrics can be reused. The conversion methods presented in Section 2.1.5 could be used to solve this problem. They provide an explicit rating by calculating the percentage of the programme that was watched by the user. I did not use them because they assume each broadcast item is a unique programme, so I suggest an alternative that takes advantage of the relation between a programme and its episodes.

For each user item pair (U, I) consider those items who had at least one watched episode E. It is considered an episode was partially watched if the user has viewed at least 5 consecutive minutes of it. This allows to discard watching sessions that represented channel surfing.

For each watched item, consider the list K of all the episodes broadcast from the first watched episode. Then sum all minutes watched from the first watched episode and divide it with the sum of the programmes duration. The value is multiplied by 10 to give a rating in the desired scale. The following formula represents this method:

$$r_{u,i} = 10 \cdot \frac{\sum_{e \in K} e_w}{\sum_{e \in K} e_d}$$

Unlike other methods, episodes broadcast before the first time the user watched the programme are not considered because it would have a negative impact on the rating for long running series. For instance, if a user never watched a long running series, he might not know the programme was aired or he simply didn't use the service before.

If the user watched a whole episode of that show, we could infer he liked it, even if he didn't watch the previous shows.

Another possibility would be to only count the watched shows but that would mean the system thought the user liked the programme even if the user never watched new episodes of it. If we count all the following programmes, the calculated rating would

drop with time, lowering suggestions of episodes of the same and related shows. We can say with more confidence the user did not like the show after he first watched than if he never watched at all. So, a rating value is not fixed, it evolves with time.

3.2.3 Recommender Module

This component is responsible to process the structured user and ratings data and return a list of recommendations personalized to the user. There are multiple algorithms that can be used as part of this process. The used and evaluated algorithms are presented in this Section.

3.2.3.1 Techniques

There are multiple algorithms a recommendation system can use to generate its recommendations, as shown in Section 2.1. In this section I list the approaches tested and used in the developed recommendation system. Most techniques are based on previous work while others were developed in this thesis with this specific domain in mind. Most techniques assume explicit feedback and take advantage of the mapping function developed in Section 3.2.2.1.

Baseline Functions

These methods were introduced in Section 2.1.2.1, and are generally used as a fallback when the other algorithms aren't able to provide personalized recommendations.

Item Mean Rating, described previously at Section 2.1.2.1, returns the item's mean rating. This function is used as a baseline for item based collaborative algorithms. A similar function, named User Mean Rating, that returns the user's average rating is also tested in a similar way. Finally, the hybrid Item and User Mean Baseline was tested and compared.

Collaborative-filtering Algorithms

The collaborative algorithms listed here were introduced in Section 2.1.2 and they are typically used when items have been previously rated by different users. The first type of techniques that were evaluated are the User-User and Item-Item Memory algorithms from Section 2.1.2.2. These algorithms can be customized with different similarity and ratings normalizer functions. Those variations were also evaluated.

The second family of techniques used is the FunkSVD Matrix Factorization family of techniques introduced in Section 2.1.2.3. Multiple iteration sizes and speeds were tested.

Finally, the Slope One techniques, introduced in Section 2.1.2.4 were tested. Both generic Slope One and Weighted Slope One algorithms were evaluated as a possible alternative to the previous collaborative filtering algorithms.

Content Filtering Algorithms

Content filtering algorithms, Section 2.1.3, are best used when we have or can extract metadata from items.

Item Vector Scorer is a content based recommender module[1] for LensKit. It was tested with content categories, tags, directors and actors.

Other techniques not based on previous research were developed in this thesis, for the purpose of testing specific ideas about the television domain. Those techniques are described here:

Item Broadcast Time

The objective of this technique is to recommend programs broadcast when the user typically watches television. The inverse function allows to increase less expected recommendations by the user, trying to improve the system's serendipity. This is considered to be a content filtering algorithm because it is based on the broadcast time of the items. It calculates the item rating depending on the broadcast date of its episodes. It first creates a representation of the user's time habits and then it uses the representation to calculate the programme rating.

For each user, there is a user time vector that represents the likelihood the user watched TV at a given minute of the day. Assuming the user watched TV for n days and a day contains minutes m, we have a function $w_u(d, m)$ that represents if a user was watching TV on a given minute of a day

$$w_u(d,m) = \begin{cases} 1 & if watched \\ 0 & else \end{cases}$$

We can calculate the chance the user watched TV at a given minute of the day with:

$$W_u(m) = \frac{\sum_{d=1}^{k} w_u(d,m)}{k}$$

Where k is the total number of days considered. Finally, we calculate the rating the user would give to the programme by calculating the average of the chances the user watched TV during the period the programme was broadcast:

$$R_{u,i} = 10 \cdot \frac{\sum_{m=s}^{s+l} W_u(m)}{l}$$

s is the programme start minute and l is its length. This formula can also be used with different time periods, like considering the weekend and week in separate when doing the calculations.

[1] Available at `http://eugenelin89.github.io/recommender_content_based/`

Item Broadcast Channel

This technique recommends programs based on the channels the user typically watches. To calculate it, we sum the number of minutes thr user watched the channel where the programme is broadcast and divide it by the total watched minutes.

Item Episode Mean Rating

User ratings for previously rated programmes change over time, since a program is made by several episodes. This algorithm analyses the user history regarding the rating of previous episodes then it predicts the rating of new episodes using a weighted average to calculate the overall program rating.

In Section 3.2.2.1 an implicit mapping algorithm was defined. It was also defined the list $K_{u,i}$ containing the user u ratings for all the episodes of item i broadcast from the first watched episode. For this algorithm we define an extension, $Kp_{u,i}$, that adds to $K_{u,i}$ the predicted ratings for the episodes broadcast in the target period.

The predicted value is calculated with the following formula:

$$P_{u,i} = \frac{1}{2^{k-1}} K_1 + \sum_{n=2}^{k} \frac{1}{2^{k-n+1}} K_n$$

K_n is the n-th rating on the list $K_{u,i}$. This formula uses the values from the previous episodes to calculate the predicted value to the next episodes, where more recent values have a greater influence on the final prediction.

The result of $P_{u,i}$ is assigned to the new episodes in $Kp_{u,i}$. Then this list is used as an input to the implicit mapping algorithm explained earlier, outputting a rating for the item.

3.2.3.2 Hybrid Techniques

As shown in Section 2.1.4, a recommendation system is generally composed of multiple techniques connected to improve the overall system quality. In this Section I present two different approaches to hybrid recommenders that connect the techniques from the last Section. These hybrid techniques were built after evaluating the individual techniques so I had the required information to know how each recommender could complement the other. The evaluation results are presented in chapter 4.

Mixed Recommender

A recommender based on a mixed hybridization method. It starts by running the first recommendation method. For the items that were not covered by it, it then runs the next recommender. It fallbacks to the next algorithm until every item was processed or until it reaches the last algorithm. The algorithms were chained as follows:

1. Item Episode Mean Rating
2. User-User Collaborative Scorer (with Baseline Subtracting Normalizer and Cosine Similarity)
3. User Item Baseline Mean Rating

The algorithms with the best evaluations are the first in the list and the ones with the best coverage are in the end. Overlapping coverage algorithms should be avoided.

TV Hybrid Recommender

This technique is composed by a set of heuristics that employ weighted, switching and mixed hybridization methods. The objective was to make a recommender that used a specific algorithm for each different situation:

- Recurring Programme
  - User has watched it before
    - Item Episode Mean Rating
  - User has never watched it before
    - Item-Item Collaborative Filtering
    - Item Broadcast Time
- New Programme
  - Item Vector Scorer
  - Item Broadcast Time

3.2.4 Explanations Generator

The explanations generator module receives a list of recommendations from the recommender module and organizes it in multiple categorized lists, providing a personal explanation for each recommended item. To establish the explanation objectives and style, the guidelines introduced by [43] and presented in Section 2.2 were followed. Then an algorithm that solved the requirements set in the first step was built.

3.2.4.1 Designing Explanations

Explanation Aims

The first step to provide explanations for this recommendation system was to choose which aims presented in Section 2.2 would be more appropriate to the TV domain and to the objectives of the system. I've listed the impact each aim would have in the system and then chose the most suitable ones.

- **Transparency** - Although it is interesting to allow the user to see how the system works, this isn't a critical recommendation system and as such this system can be implemented either as a black box or a white box model.

- **Scrutability** - Allowing users to tell the system it is wrong is important since it makes assumptions of the user tastes with the collection of implicit feedback
- **Trust** - User trust is important if we want the user to come back to the system.
- **Effectiveness** - An effective system is not necessary because users don't have to pay or consume the whole item. It is not critical as a decision to choose where to go on vacation.
- **Persuasiveness** - Persuasive systems increase the number of items the user buys, but it can also decrease the users trust in the long term. This is not an important aim since this system doesn't include items the user must pay to watch.
- **Efficiency** - One of the identified problems people have with current systems is they take some time to find something to watch, as such efficiency is important to this system.
- **Satisfaction** - As identified previously by other authors, satisfaction in an entertainment application is very important.

After listing all the aims, those which seem more important for this project are Satisfaction and Efficiency. The other aims are more important when applied to more serious systems that don't target entertainment or require the user to make more serious or expensive decisions.

Presentation and Interaction

The next step was to choose the adequate presentation and interaction methods for recommendations and explanations, detailed in Sections 2.2.2 and 2.4. For this recommendation system, the most adequate presentation technique for explanations is the top-n items recommendation because the underlying system already provides an ordered list of recommended items. The recommended items are organized in multiple top-n lists, grouped by a common property, and each item has an individual text description that tries to improve the explanation.

User interaction with the recommendation system is limited to the interaction methods a television set provides. Users do not like to navigate or input text on their televisions. As such, typical interaction like listing the user preferences and rating the items is limited, and the system should be able to provide explanations without feedback.

Explanation Styles

The final step on explanation design was to choose the explanation style, as presented in Section 2.2.3. Collaborative and Content-based explanations were chosen because the underlying algorithms and data support the required information to generate an explanation.

For each item, an explanation consisting of a title and a subtitle will be generated. The objective is to generate a specific title and subtitle that describe the items, tailored for the user. Items are grouped by title, and each subtitle details the item individually.

The following table lists the titles generated by the explanations algorithm.

Explanation Style	Type	Format	Example
Collaborative	Popular	Popular Items	Popular Items
Collaborative	User	People like you also like	People like you also like
Collaborative	Item	Fans of [item] also like	Fans of Breaking Bad also like
Content-based	Actors	Programmes with [actor]	Programmes with Andrew Lincoln
Content-based	Directors	Programmes directed by [director]	Programmes directed by Christopher Nolan
Content-based	Year	Programmes from [year]	Programmes from the 1980 decade
Content-based	Channel	Programmes broadcast on [channel]	Programmes broadcast on FOX Movies HD
Content-based	Category	[category] Programmes	Sports Programmes

Table 3.2: Possible category titles

These titles explain to the user why and how the items are grouped and in some situations they also present a hint to why they were shown to the user.

The generated subtitle tries to clear why each individual item was presented:

Explanation Style	Type	Format	Example
Collaborative	Popular	Popular Item	Popular Item
Collaborative	User	People like you also like	People like you also like
Content-based	Same Item	Because you watched a previous episode	Because you watched a previous episode
Content-based	Similar Item	Because you watched [item]	Because you watched Breaking Bad
Content-based	Actors	Programmes with [actor]	Because you like Andrew Lincoln

Table 3.3: Possible item explanations

3.2.4.2 Explanation Algorithm

The algorithm for the explanations design presented in the previous Section is based on feature vector manipulation.

The input for the explanation algorithm is the list of predictions generated for the user by the recommendation module, and the item's metadata.

1. **Build Vector Feature Space**
   For each item property available in the items provided by the recommender module, add to a list representing the vector features space. Those properties are based on the types listed in Table 3.2.

   $$V_{fs} = \{Actor_1, Actor_2, \ldots, Category_1, \ldots\}$$

   This vector space can be used to represent any kind of item, group of items and users of the system.

2. **Generate Item Vectors**
   For each item, make a feature vector and assign each item property to it.

3. **Generate Category Vectors Lists**
   Generate all valid combinations of category vectors. Iterate over all item vectors and create combinations of the defined properties. Group, without duplicates, the combinations from all items.

4. **Generate User Vector**
   Generate a user vector based on the items predicted by the recommender module. For each predicted item, sum its vector multiplied with the prediction score

   $$\overrightarrow{U_u} = \sum_{i \in I} (r_{u,i}) \overrightarrow{I_i}$$

   $\overrightarrow{U_u}$ is the user vector, I is the list of recommended items, $r_{u,i}$ is the predicted rating and $\overrightarrow{I_i}$ is the item's vector.

5. **Calculate similarity between User and Categories**
   For each category, calculate the cosine similarity between the category vector and the user vector, then sort the results in descending order.

6. **Assign items to Category**
   Sort all programmes by its predicted rating, in descending order. Assign each programme to the first category that contains one of its properties. After that, cycle through all the lists and delete those lists with less than 3 programmes, and truncate the lists with more than 20 items. Repeat the algorithm to reassign the removed programmes. Repeat until all programmes are assigned.

3.2.5 Recommendations Dispatcher

This module is responsible for caching and refreshing user recommendations. It runs the recommendation and explanation modules and saves the results locally so the system is always prepared to serve recommendations. There are two triggers used to refresh the recommendations:

- Once per day, during the night, the system refreshes all user recommendations. In most cable TV operators the EPG and items available to the user change every day, so the system has to run once a day to always have updated recommendations.
- When a user is watching a programme, update the user recommendations so they reflect the updated user profile.

3.2.6 Web Services

Web Services connect the Frontend application to the recommendations dispatcher module. For this project, only the action to request the recommendations list is necessary.

3.2.7 Recommendations Frontend

The final step on the development of this project was to develop a User Interface that allows the users to see and try recommendations. This interface is presented to the users as part of the user evaluation phase. The objective was to make an interface the users felt familiar that could be connected to existing set top box and Smart TVs interfaces, based on the guidelines presented in Section 2.4. The interface presents the recommendations and their respective explanations.

3.2.7.1 Recommendations Screen

This screen, shown in Figure 3.2, is the main menu of the recommendation interface. It shows personalized recommendations and explanations to the user. It follows the typical structure TV applications use to show lists of multimedia items: Items are grouped by categories. The category title is shown above the list of items. Each row represents a category. Each item is represented by an image that depicts the item content. The item title and explanation are shown below the currently selected item.

Navigation between items is done using the device remote control. Navigation between items of the same category is done using the left and right keys, while navigation between categories is done by pressing the up and down keys. To select and preview the item details, the user should press the enter key.

Figure 3.2: Recommendations Screen

3.3 Implementation

This Section shows the implementation details of the modules described in the previous Section.

3.3.1 Data Sources and Data Preprocessor

The Data Preprocessor module was implemented as a Java Interface that provides multiple methods for data interaction with the structures defined on Table 3.1. Each one of those structures were also implemented as Java classes. The following table shows the available methods presented by the data preprocessor for interaction with the data:

Data Type	Methods
User	list system users, get user details, get user ratings
Programme	list available programmes, get programmes properties, get user ratings
Properties	list programmes
Ratings	list explicit ratings, list implicit raw ratings

Table 3.4: Methods available for interaction with data

The objective is to extend this interface and abstract data source differences. For this project, two different implementations of this interface were built: one for the offline evaluation data set, presented in Section 4.1, which was based in CSV files, and another one for user studies, based on Neo4j databases.

3.3.2 Recommender Module

The base of the recommender module is the Lenskit Framework. The data preprocessor is connected to this component, using a custom LensKit Data Access Object (DAO), their interface for data collection.

Recommendation algorithms, in LensKit, are implemented as a ItemScorer subclass. An ItemScorer is an interface that defines methods where you provide a user and a list of items to predict, and it returns a vector with the predicted score for each item. The list of classes available in LensKit is shown in Table 3.5.

Algorithm Name	LensKit Scorer Class
Baseline Functions	
Item Mean Rating	ItemMeanRatingItemScorer
User Mean Rating	UserMeanItemScorer
Collaborative-filtering Algorithms	
User User Scorer	UserUserItemScorer
Item Item Scorer	ItemItemScorer
FunkSVD	FunkSVDItemScorer
Slope One Scorer	SlopeOneItemScorer
Weighted Slope One Scorer	WeightedSlopeOneItemScorer
Content Filtering Algorithms	
Item Vector Scorer	TFIDFItemScorer

Table 3.5: List of used Lenskit algorithms

For the other implemented algorithms, first a model builder class was done that builds the user model, and then a custom lenskit class.

For the hybrid algorithms, a simple scorer class was built that delegated to the other scorers, depending on set rules.

3.3.3 Explanations Module

The explanations module was implemented as a Java class without any dependency to the LensKit Framework, except for its Sparse Vector implementation for easy and fast manipulation of vectors.

3.3.4 Recommendations Frontend

The user interface was implemented as a HTML5 single page application. It interacts with the recommendation system by using its Web Services. The following HTML5 technologies were used:

Bootstrap Framework for building HTML5 responsive interfaces. Used to build the application layout.

jQuery JavaScript library for simple cross-platform DOM interactions and AJAX calls.

jQuery Transit CSS3 Animations using the jQuery animation methods, allowing to use the device GPU acceleration.

Moonstone UI An HTML5 library to build TV user interfaces. Although the full library wasn't used, some CSS components were adapted.

3.4 Summary

This chapter presents the development of a recommendation system pipeline and its modules. The three objectives defined in Section 1.2 were met. First, a recommendations engine was built and is part of the pipeline. An explanations module implements the objective of providing explanations for recommendations. This module presents an approach to improve user satisfaction not present in other recommendation systems for cable TV. Finally, a User Interface was built to show and evaluate recommendations.

Other components had to be developed to meet the mentioned objectives. The implicit feedback mapping algorithm allows to generate recommendation without requiring the user to change its behaviour when watching TV. New recommendation techniques were developed with the TV domain in mind.

In the next chapters, evaluations are performed. Offline Evaluations are used to evaluate if the developed recommendation algorithms are an improvement when compared with the other used techniques. User Studies are used to evaluate the user satisfaction in the system and are used to evaluate if it improves over their cable TV providers recommendation systems.

Chapter 4

Offline Evaluation

The first step to validate the developed recommendation system is offline evaluations. This Section is based on the evaluation methods presented in Section 2.3. This Section focus on the evaluation of the recommendation algorithms.

4.1 Dataset

In Section 2.3.2 a list of public datasets was presented. These datasets have been used extensively in research and even used to evaluate algorithms in the TV watching domain. They belong to a similar but different domain, the movies domain. Although similar, the TV domain has important differences such as the predominance of implicit rating data and the time restrictions in the items consumption. As such, there is the need to find or build a more appropriate dataset. As far as I am aware, as of the publication of this thesis, there is no public dataset that satisfies the requirements presented in Section 3.2.1.

4.1.1 TV Ratings Dataset

TV ratings are usually measured by installing devices in voluntary households that collect TV watching data. This information together with an archive of the broadcast programmes can be used as a dataset that fulfils the requirements.

I was able to get a portion of data originally collected for the measurement of Portuguese TV ratings. It covers Portuguese free-to-view TV channels for a time span of six months. It contains a basic user profile, a list of broadcast programmes and user watching sessions. We can extract the implicit data which makes it a good candidate to an evaluation dataset. A summary of the data in this dataset is available on Table 4.1.

Type	Total
Users	2,071
Items	1,640
Episodes	17,264
View Sessions	2,148,035
Channels	4

Table 4.1: TV Ratings Dataset count

Items are organized and ordered like an EPG. Each channel has a list of programmes ordered by their start date. Each programme is associated to an item by its title. An item is a group of programmes that share the same title, like multiple episodes of a TV Series. Each item contains the basic details required for basic content-filtering techniques. They contain a title and subtitle and a hierarchy of up to three nodes identifying the category.

Field	Example Value
channel	3
start_date	1996-01-08 22:51:14
duration	4573
title	Noite De Estreia
subtitle	Pulp Fiction
main_category	Ficção
sub_category	Filme
third_category	Suspense

Table 4.2: Example item

A user watching session consists of information on the periods a user was tuned to a channel. This information in association with the items broadcast data can be used to find how long each user watched each item.

Field	Example Value
user_id	17382723
channel	3
start_date	1996-01-08 21:54:00
end_date	1996-01-08 22:51:00
duration	61

Table 4.3: Example session of a user that watched a portion of the example item

4.1.2 Assembled dataset

The TV Ratings dataset presented in the previous Section doesn't contain detailed metadata, so content-filtering algorithms can't be completely tested. To fully test the presented algorithms, I needed to use another source for the items metadata. An assembled dataset, nicknamed "TV Provider Dataset", was produced. It consists of the TV Ratings Dataset with collected metadata from IMDB. The data collected from IMDB is the creation date, directors and participating actors.

4.1.2.1 Item Mapping between data sources

A simple mapping algorithm was defined to map items between the two data sources. IMDB provides a search function that allows to search program titles in Portuguese.

- For each item in "TV Ratings Dataset"
  - Search IMDB for programmes with the same title
    - For each returned item
      - If the programme date is less than or equal the broadcast date and it is of the same category (Movie or TV Series), use this item.

4.2 Evaluated Measures

Each algorithm will be evaluated and compared with the same set of measures. The first evaluated measure will be the item's coverage. It is the percentage of items the algorithms were able to produce a prediction. An algorithm might not be useful if it isn't able to produce predictions for most of the items in the system, even if it scores well on other measures. This means the algorithm would have a low recall.

RMSE is one of the most popular measures for recommendation systems evaluation. It will be used to compare the predicted user ratings to their actual rating. The Netflix RMSE implementation will be used, which means the result is adjusted to the 0 to 5 scale. Two variants of this measure will be used. First we define Global RMSE, that returns the RMSE calculation for all items, including those not covered by the algorithm. The other measure, Coverage RMSE, only covers the items predicted by the algorithm. Global RMSE allows one to compare between all algorithms, while the coverage RMSE is better to compare between algorithms of the same type.

Finally, the time the algorithm takes to produce recommendations will be measured. If an algorithm takes too long to generate predictions, they might be outdated when presented to the user.

Algorithm	Global RMSE	Coverage RMSE	Coverage	Time (s)
Baseline Functions				
Item Mean Rating (I MR)	1.331	1.331	**100.0%**	**3.57**
User Mean Rating (U MR)	1.316	1.316	**100.0%**	**10.45**
User and Item Mean Rating (U/I MR)	1.287	1.287	**100.0%**	10.85
Collaborative-filtering Algorithms				
User User (U/I MR, Gaussian, Cosine)	2.294	0.963	71.2%	147.34
User User (U/I MR, Gaussian, Pearson)	2.301	0.983	71.2%	133.09
User User (U/I MR, Baseline Norm., Cosine)	2.277	**0.912**	71.4%	222.89
User User (U/I MR, No Normalizer, Cosine)	2.309	1.005	71.4%	126.10
User User (U/I MR, Decoupling, Cosine)	2.291	0.953	71.2%	150.79
Item Item (I MR, Baseline Norm., Cosine, 20 Neighb.)	2.306	0.997	71.4%	25.37
Item Item (I MR, No Normalizer, Cosine, 20 Neighb.)	2.344	1.100	71.4%	16.97
Item Item (I MR, Baseline Norm., Pearson, 20 Neighb.)	2.301	0.984	71.4%	29.47
Item Item (I MR, Baseline Norm., Pearson, 10 Neighb.)	2.305	0.994	71.4%	22.79
Item Item (U/I MR, Baseline Norm., Cosine, 20 Neighb.)	2.301	0.984	71.4%	23.20
Item Item (U/I MR, Baseline Norm., Pearson, 20 Neighb.)	2.297	0.972	71.4%	28.99
Item Item (U/I MR, Baseline Norm., Pearson, 10 Neighb.)	2.298	0.974	71.4%	22.99
FunkSVD (Features=10; Iterations=125)	2.304	0.993	71.4%	21.32
FunkSVD (Features=25; Iterations=125)	2.294	0.965	71.4%	21.93
FunkSVD (Features=25; Iterations=500)	2.310	1.008	71.4%	51.34
FunkSVD (Features=50; Iterations=500)	2.295	0.971	71.4%	96.90
Slope One Scorer	4.656	1.640	49.8%	22.12
Weighted Slope One Scorer	4.657	1.654	49.8%	37.43
Content Filtering Algorithms				
Item Vector Scorer	2.092	2.058	93.8%	11.90
Item Broadcast Time w/Channel	2.363	1.770	71.2%	17.17
Item Episode Mean Rating	2.873	**0.214**	21.7%	13.18
Hybrid Techniques				
Mixed Recommender	**1.012**	1.012	**100.0%**	200.00
TV Hybrid Recommender	**0.963**	0.963	**100.0%**	34.08

Table 4.4: Evaluation Results
The highlighted rows represent the algorithms produced in this thesis and bold values represent the best values for each metric.

4.3 Results

The evaluation results are shown in Table 4.4. In this Section the presented results are discussed and the algorithms are compared.

4.3.1 Baseline Functions

The first three evaluated algorithms are baseline functions. All three algorithms were able to get full item coverage and so their global and coverage RMSE values are equal. The baseline algorithm with the lowest error value was the User and Item Mean Rating. Mixing both the user and item means is useful because they complement each other. Although the RMSE value is similar, the second best baseline algorithm was User Mean

Rating. It means the user mean is better than the item mean, when predicting an item.

All algorithms are relatively fast, with the Item Mean Rating being the fastest.

4.3.2 Collaborative-filtering Algorithms

Most of the evaluated algorithms belong to this type of approach. Every algorithm, except the ones based in Slope One techniques, present similar coverage values.

Both Slope One and the Weighted Slope One algorithms present the worst values of this category and the overall system, both on the RMSE and the coverage metrics. They present an error even worse than the baseline. Their authors [28] say they are generally comparable with the other collaborative algorithms, so it might not work as expected in the TV domain.

The other evaluated algorithms generally had an RMSE value lower than 1 for covered items, with a few exceptions. The algorithm with the lowest error was the User-User Collaborative-filtering variant, with User and Item Mean as baseline, Cosine as the similarity function and the Baseline Normalizer function. Unfortunately, it also was the slowest evaluated algorithm. In general, algorithms that used Cosine similarity presented better values.

FunkSVD presents similar values when compared to the User-User algorithms but it is much faster. The best variant was the one with 25 features processed in 125 iterations, the values suggested by the LensKit documentation. More iterations and features didn't improve the overall score.

The last algorithm family in this Section is the Item Item Collaborative filtering. They generally presented worse values than the previous two algorithms but not significantly. Generally we found out that 20 neighbours is a better value than 10, having better scores. I also found out that Pearson Similarity performed better than Cosine Similarity for Item Item algorithms, unlike User User algorithms.

All algorithms in this Section had a worse global RMSE than the baseline functions, although significantly better coverage RMSE values.

4.3.3 Content-filtering Algorithms

Item Vector Scorer was the only tested content filtering algorithm previously implemented for Lenskit. It has a high coverage but it has a high error value compared to most of the other algorithms. The evaluated collaborative filtering algorithms are better in finding the probable user rating than this.

The other two algorithms evaluated in this Section were developed in this thesis. The first is the Item Broadcast Time. Although better than the previous algorithm, the RMSE value is higher than the baseline methods, meaning the date and channel the items are broadcast is not a good indicator of the observed user ratings.

The other algorithm is the Item Episode Mean Rating. It has a higher coverage RMSE when compared to the other algorithms, but the lowest coverage, meaning that it can only generate predictions for a smaller set of items.

In terms of speed, these algorithms were almost as fast as the baseline ones.

4.3.4 Hybrid Algorithms

Using multiple techniques in a recommender is a clear advantage. These algorithms present a complete coverage and they also present the best global RMSE values.

The Mixed Recommender has the second best Global RMSE. Most of the other algorithms have a better coverage value. It is also very slow.

TV Hybrid Recommender has a good coverage RMSE when compared to other algorithms and it is also relatively fast, being a good candidate for the next step of evaluations.

4.3.5 Discussion

This step was important to understand the differences between the algorithms and how they would fit in a production recommender system. Testing each algorithm individually allowed to create two proposals of hybrid techniques. One of which has a greater RMSE when compared to the individual techniques, and full coverage.

Sometimes developing algorithms with the target domain in mind makes sense, like the ”Item Episode Mean Rating”. This algorithm is not generic like the others, but understanding the specific domain allowed to achieve low values of RMSE, although with low coverage.

It is interesting to see how most techniques had a coverage RMSE around 0.9 to 1.0, and how difficult it is to reduce this value without having slower algorithms or reducing the coverage. The user's behaviour is unpredictable and so it is not easy to have a model that accurately predicts the user rating.

Offline Evaluation was good to easily compare the algorithm performance when predicting what the user watches without the influence of a recommendation system, but we can't really test its feasibility without including direct user interaction. In the next chapter user studies were performed and the system was evaluated, to see the impact this system had on user behaviour and experience. The algorithm that was chosen for this step was the TV Hybrid Recommender.

Chapter 5

User Studies

5.1 Objectives

In this stage of the project, I developed and performed user evaluations to further validate the work developed in the thesis. As presented in Section 2.3, user studies allow to test for changes in user behaviour when in front of a recommendation system and they also allow to directly ask them questions and feedback on the system.

The objectives of this stage were:

- Understand and validate the TV consumption habits detailed in Section 1.1 and the user interest for TV recommendation systems.
- Find out the type of TV recommender system the user prefers.
- Evaluate the quality of generated recommendations and explanations for the users.
- Determine the user interest for each type of explanations contemplated in the system.
- Validate the targeted explanation aims.
- Gather user opinions and interest on the developed recommender system.

5.2 Evaluation Method

Due to the nature of this project, it was necessary to find users willing to have their television consumption habits recorded, so it would be possible to generate and test personalized recommendations. It was also necessary to have a dataset that matched the time period and channels the users watched. These questions were solved after recruiting users and after assembling a dataset using public EPG services available on the internet.

The users' television usage was recorded for a month, and then the recommender generated a list of predictions for the week after.

To test the recommendation interface, the objective was to mimic the TV user experience. A computer running a HTML5 browser was setup with an infrared receiver. This allowed to control the interface with a remote control, emulating the TV experience. Before the experience, users were asked what was their cable TV provider, allowing us to configure the same TV remote they usually use.

5.2.1 User Evaluation Session

Before starting the evaluation, I introduced the user to the subject of this thesis and to the application the users were going to evaluate.

The first step was to ask demographic questions about the user, like gender, age and academic qualifications. The user was also asked about how many people watch TV in their homes. During the evaluation process, the user did not have to input the answers, the interviewer filled the answers instead. Then the user had to answer questions about their TV consumption, like the frequency they watch TV and how long they generally take to find something to watch. Finally, before interacting with the system, they answered some questions about the recommendation system of their TV provider and their interest in recommendation systems in general.

After this step, the user is asked to freely interact with the recommendation system for the first time and their recommendations. The interviewer then asks the user to rate the categories, the recommended items and the explanations, for the first twelve categories. During this step, the participants were asked to think out loud, and the results were annotated by the interviewer. In the end of this step, the users were asked if any of the categories below, the rated ones, should be better positioned in the list and also to give general comments about the system.

After this, the user had to rate some sentences about their perception of the system, like rating the quality of the recommendations and their satisfaction using the system. Finally, the user had to rate their interest in the explanation categories and descriptions, presented in Section 3.2.4. The evaluation session ends with a final comment from the user.

The full questionnaire is available in appendix A.

5.3 Results

5.3.1 Sample Characterization

13 users from 8 different households participated in this evaluation. Every household had a Cable TV subscription. 46% of the users were aged between 18 and 24 years and 30% were over 34 years old. Most people were graduated, 62% of people had a Bachelor's degree. Only one user was still graduating. 92% of the people had at least 3 family

members at home that watched television. 85% of the users watch television every day, and the other users said they watch TV every week. When asked if they watched TV with their family, most answered at least once per week.

About 54% of the users say they take between 5 to 10 minutes before finding a programme to watch. The most mentioned used method in this process is channel surfing. Then, users mentioned using the time shifting capabilities of their cable TV provider to watch new episodes of their favorite programmes.

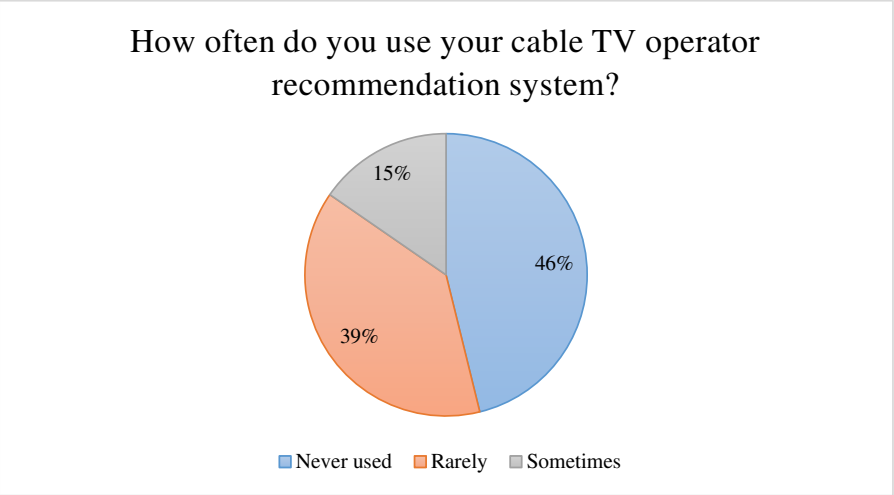

Figure 5.1: Results for question ”How often do you use your cable TV operator recommendation system?”

When asked if their current cable TV provider had a recommender system, 84% answered positively. The other users weren’t sure. 46% of the users said they never used it and only 15% said they use it sometimes, as seen in Figure 5.1.

When asked about their satisfaction in their cable TV recommender system, everyone that answered rarely in the previous question, also answered rarely in this question. The users that answered sometimes, were satisfied most of the time.

Users were asked if they thought recommendation systems and explanations are important. On Figure 5.2 it can be seen that people find favourable the existence of recommendation systems and are very favourable to explanations.

For the last question the objective was to find if users preferred either a more conservative or a more serendipitous recommendation system. The results can be seen in Figure 5.3. This question didn’t have a middle option on purpose, so the user had to choose a side. Most users selected values near the middle, with most of them choosing slightly more conservative values.

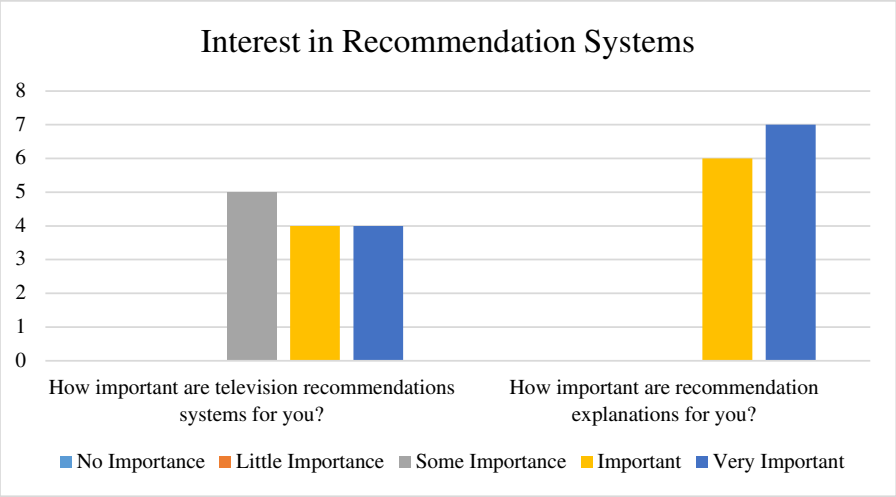

Figure 5.2: Results for the questions about recommendation system usage

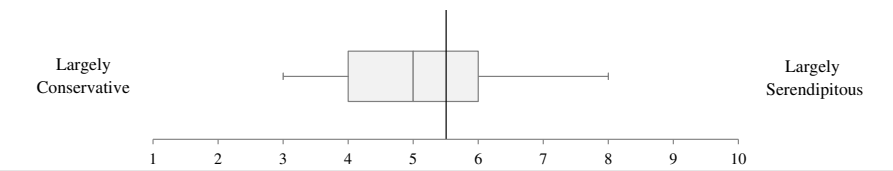

Figure 5.3: Results for the question about recommendation system type

5.3.2 Session

During the interaction with the developed recommender system, users had to navigate through the suggestions and rate the presented categories, programmes and explanations. To generate these recommendations, the complete recommendation pipeline was used. The recommendation algorithm in use was TV Hybrid Algorithm and the system was run with the user watching history data collected earlier. It should be noted again that the categories and programmes were presented to the user in a personalized order. Here we are evaluating the capability of the system to present items the user likes first and not the user interest on a particular category.

I asked the users to rate their interest for the top twelve categories that were presented to them. Every user liked at least eight of the presented categories. On the graph 5.4 we can see how the users liked each presented category. The users tend to prefer the categories presented first, with the first category chosen by the system being unanimously approved by them.

Then the users were asked to rate the individual items and explanations. This allowed me to validate the previous result. The results are shown in Figure 5.5. On the left chart we have the percentage of items liked by category position. The categories that appear first have better results. In the right chart, we can see the percentage of items liked by its

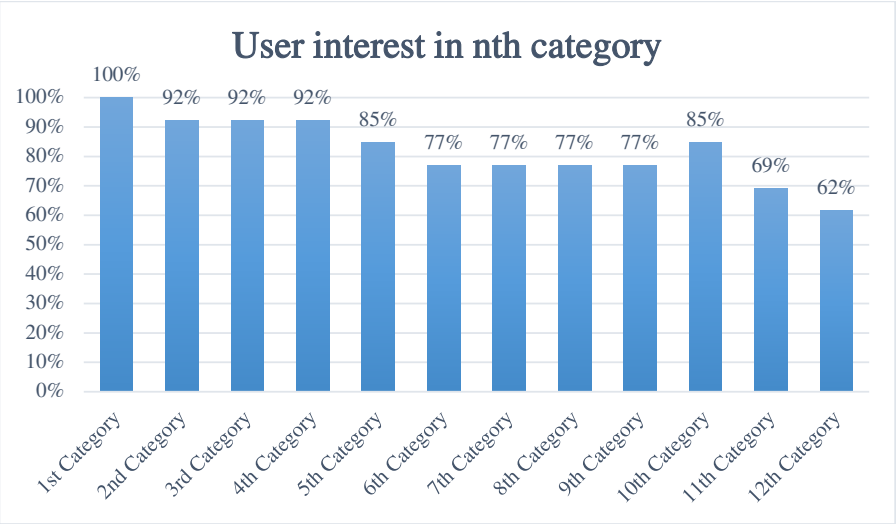

Figure 5.4: User interest by category position

position in the category. The items are shown in groups of four. The items that appear first have a better rating than the others.

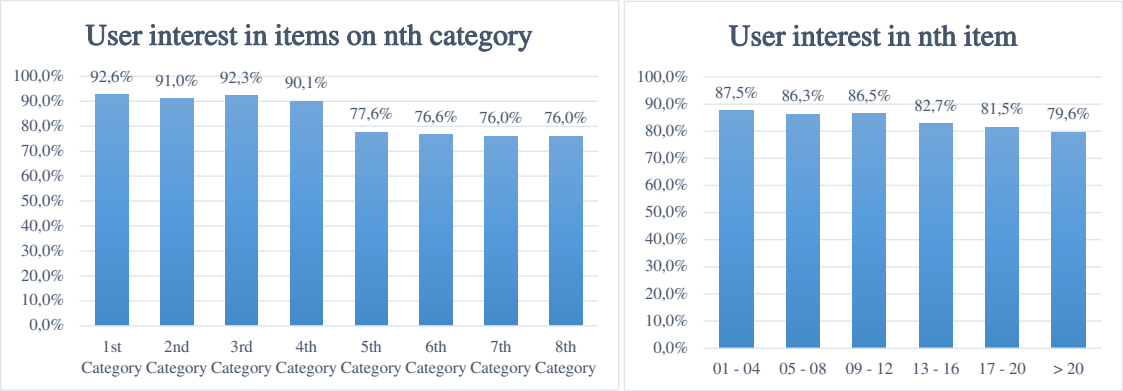

Figure 5.5: User interest in the presented items

5.3.3 User Feedback

After using the system, users were asked some questions regarding their opinion on what they interacted with. As presented in Figure 5.6, in a scale of 1 to 10, on average, people rated 7.54 for the recommendations quality, 6.85 for the categories order and 7.77 for explanations quality. Most people were satisfied with the overall quality of the system and the explanations quality. One user was not satisfied with the order of the presented categories, and rated it 4. In Figure 5.7, the results of the sentences the user had to rate regarding their opinion on the system are shown. In terms of usability users agreed that the system is clear, efficient, and they are satisfied when they use it. Most people felt

they were more likely to watch programmes they usually didn't watch. Some people felt explanations were important, but there were more users that felt otherwise. When asked about the reasons, one user said most recommended programmes were self-explanatory, so it wasn't necessary to look at the text.

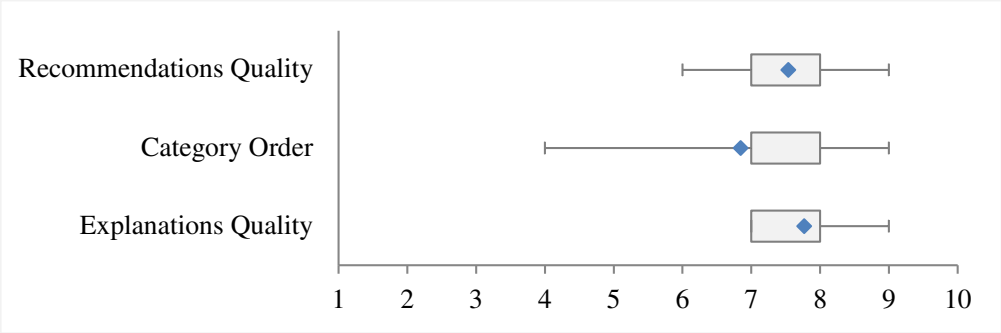

Figure 5.6: User feedback

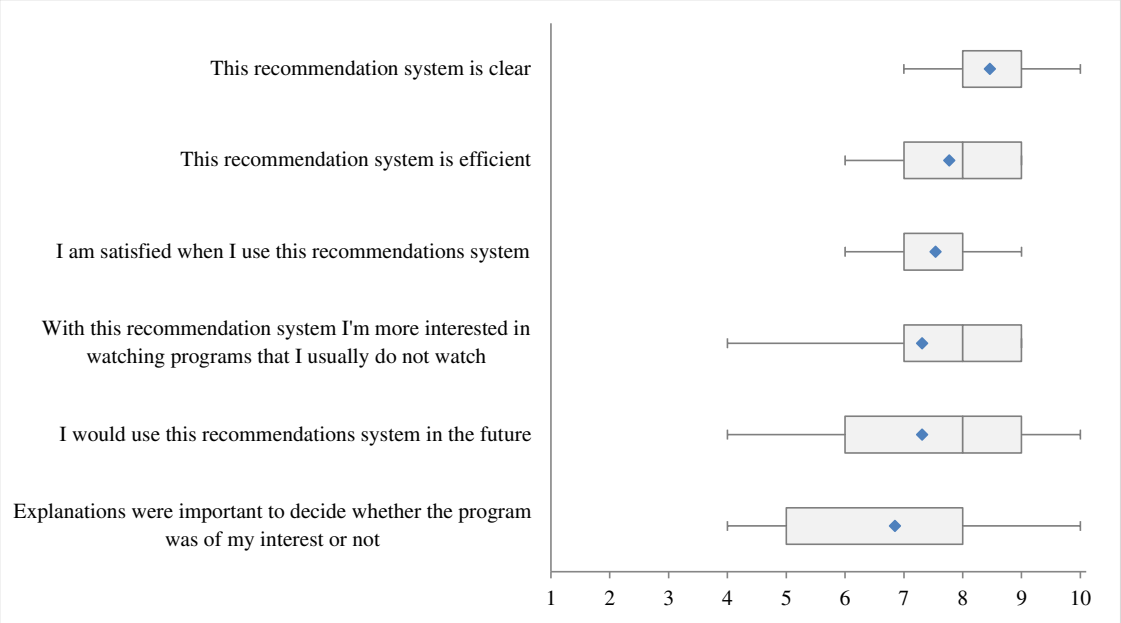

Figure 5.7: User sentences

5.3.4 Explanations

Users were asked to rate the available explanation categories and descriptions. The results are presented in Figure 5.8. Most categories were liked by the users. The most liked category was "New episodes of your favourite programmes", where most people rated the maximum value. Most of the other categories were also rated positive. The category

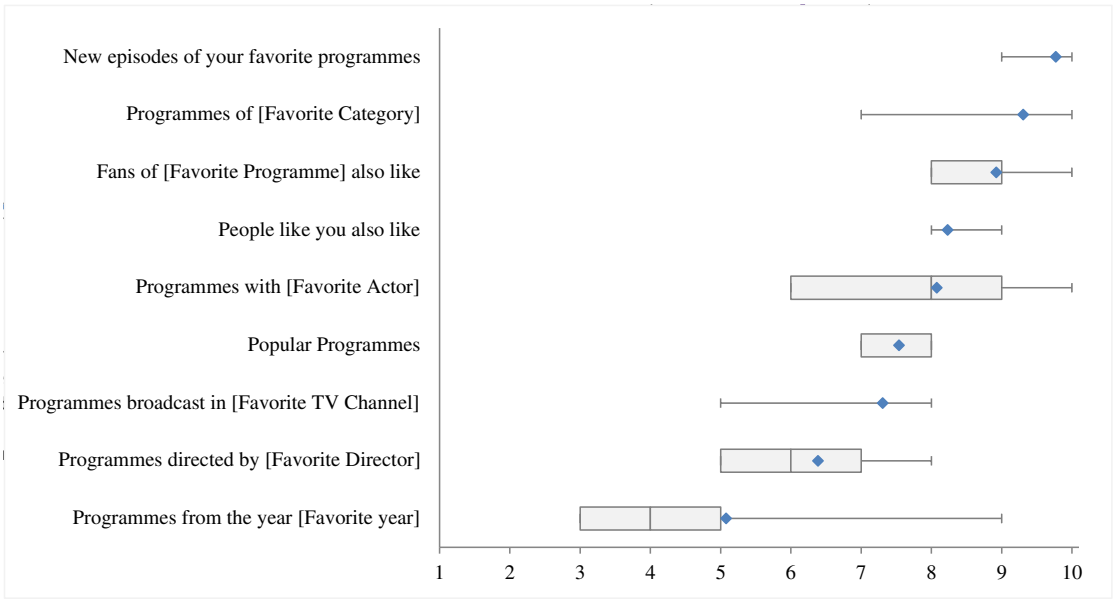

Figure 5.8: Explanation title results

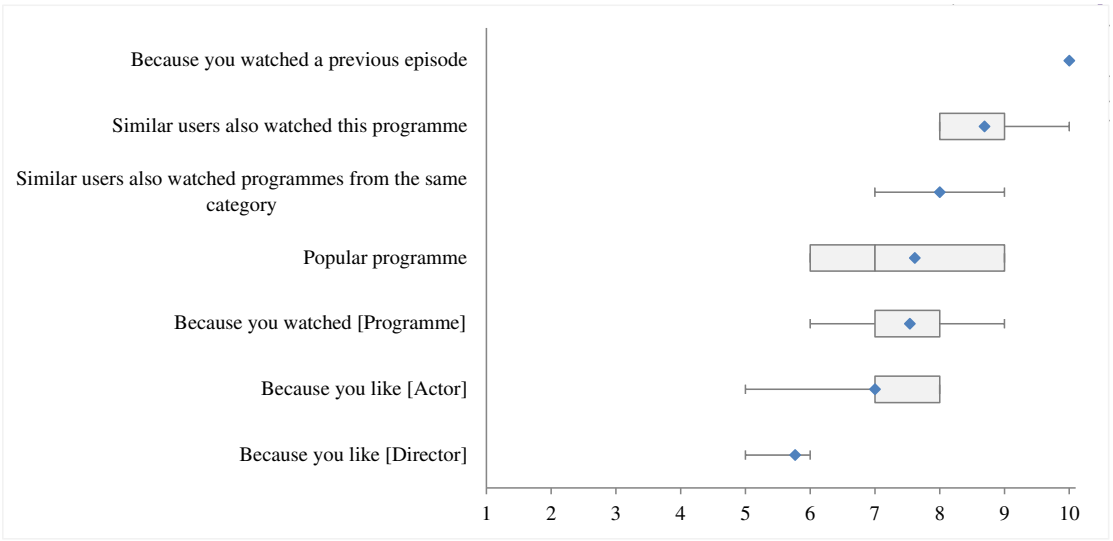

Figure 5.9: Explanation details results

people liked least was ”Programmes from the Year X”, although there were users that also gave a very favourable score to it.

When asked if they missed any explanation type, most users said they didn’t. Those who did suggest explaining a programme based on what their friends profile and another person suggested explanations based on the time slot the programme was broadcast.

Finally users were asked to evaluate the explanation types that described an individual item. The results are shown in Figure 5.9. Again, the most rated explanation was ”because you watched a previous episode”. Everyone rated the maximum score, meaning no one had doubts about its meaning. The description people liked less was the one that justified

the recommendation because the user likes the director of the programme.

5.4 Discussion

User Studies allowed to test with real users the TV Hybrid Recommender algorithm. It was the algorithm that had the best result in the Offline Evaluation step. This allowed to get user's feedback and suggestions that otherwise were impossible to get.

Here I discuss the objectives for this evaluation step, defined in Section 5.1.

Comparing these results with the studies [16] previously referenced in Section 1.1, we observe that these users don't have the same problems finding a programme to watch because on average they take less time. Most users tried their cable TV provider recommender system at least once but most didn't like it. Only a few users told they used it when looking for something to search. Nevertheless, they all felt a good recommender system was important and most were very interest in recommendation explanations.

When asked if the users preferred a more conservative or a more serendipitous recommender system, most opted for a balance between the two options, with a slight preference for more conservative systems. A result that supports this preference is that the most preferred explanation type was when a programme the user watches was shown with the caption "because you previously watched". Most users, while thinking aloud, commented recommendations on programmes they usually see improved their satisfaction, as long as they were properly labelled. It also allowed them to look more carefully to the more serendipitous recommendations. Most users answered they were more interested in watching new programmes, with this recommendation system. When asked directly about the recommendations and explanations quality, users approved the system.

Users liked most of the explanations contemplated by the system with just a few exceptions. Most did not want alternative explanations, since these covered most of the use cases. Users also validated the defined explanation aims, saying it is efficient and it makes them satisfied. Besides this positive feedback from the users, there was an interesting behaviour during the thinking aloud phase that divided the users. Sometimes the system provided an explanation that the user felt was wrong, like explaining the user liked a programme because he watched another programme he didn't watch. Some users liked the wrong explanation, because it allowed them to easily ignore the recommendation while others didn't like at all that the wrong explanation appeared.

Overall, most users wanted to continue to use this recommender system in the future. This study represented a small sample of users that interact with television. Due to the requirements of this study it is difficult to gather a larger sample. Ideally this project would be integrated in a production system and online studies would be performed, allowing to reach more users and to study the user behaviour for longer periods of time.

Chapter 6

Conclusion

This thesis studied and built a recommendation system, an area that is rapidly changing and which is currently very relevant. During the development of this thesis, the biggest Cable TV providers in Portugal released new products with focus on content recommendation[1] [2]. Nevertheless, the user study performed in this thesis allowed to see there is still much room for improvement in current recommendation systems, and this work tries to provide an alternative implementation with a significant focus on implicit user feedback. Most common algorithms were compared and tests showed they perform better when combined in hybrid recommender systems. Having two error metrics, one based on item's coverage and another based on all items in the system, allowed to better understand why the items coverage is important. Explanations are not a new concept in many current recommendation systems, but Cable TV providers seem to generally ignore this possibility. Here, an explanations component with focus on simple text-based descriptions was built and shown to be liked by its users.

Objectives

This thesis objectives were meet:

- **Recommendation Engine** A complete pipeline that allows to generate and evaluate recommendations was built. It includes a recommendation engine able to generate recommendations tailored for television, based on the user profile. Many algorithms were tested and some were created for the TV domain. Two hybrid algorithm implementations were made. After evaluations, TV Hybrid Recommender was chosen to be used and tested with real users.

- **Explaining Recommendations** A module for explanations generation was built. An algorithm for explanations was created based on a list of requirements. Users

[1] NOS renova plataforma Iris e estabelece novos standards internacionais `http://goo.gl/7k9uze`

[2] MEO revoluciona experiência de ver TV em Portugal com nova interface inteligente `http://goo.gl/YjQx2a`

liked it and it is a good differentiator to existing recommenders in cable TV providers. As it is independent of the recommendations module, many algorithms can easily be tested without affecting recommendations.

- **User Interface** A user interface to show recommendations was built. It is similar to existing interfaces but allowed to perform user evaluations. Only the recommendations screen was developed.

Challenges

In section 2.8, challenges for recommendation systems were defined. Here, I explain how the developed work has addressed some of the challenges described.

- **Implicit Feedback** This thesis proposed a new solution to the implicit feedback. Users can use the recommendation system just by watching television like they are used to. It implements a implicit to explicit mapping function. This allows to reuse the existing algorithms. Although it wasn't compared to other existing mapping functions by the users, we identified some problems that were fixed by this implementation. During the user studies step, users generally acknowledged them or someone in their families saw the identified programmes.

- **Sparse Data and Item** The developed hybrid algorithms use content-filtering algorithms that are able to generate recommendations when no user rated the item before. During user studies there was no specific test related to this, but since the user sample was reduced, most items didn't have any rating at all, meaning those recommendations were based just in content based algorithms. As said previously, users were generally favorable to the given recommendations, so the system was able to overcome this challenge.

- **Real-time requirements** A component, Recommendations Dispatcher, was built to solve this challenge. Recommendations are always cached and ready to be served to the user. New recommendations are generated to all the users when items are added or removed. Recommendations are also generated when a user is watching an item.

The other two challenges were not addressed:

- **Who's Watching** This is an interesting challenge because some users said they would rather not see recommendations for other household members. A simple solution might be having user profiles and asking the user to select a profile before they watch TV. Less pervasive methods have been researched, like using accelerometers in remote controls to identify who is watching [9].

- **Quality of Metadata** This challenge wasn't explored because it was not difficult to find and use structured information about TV programmes on the internet, such as websites like IMDB. It would be interesting if that information could be extracted automatically from the audio and video streams but such approach would be out of the scope of this thesis.

Future Work

This thesis focused on a subset of the much larger recommendation systems investigation area. Due to the time limitations of a thesis, it was not possible to further detail some mentioned topics.

There are many algorithms and alternative approaches to recommendations that could have been tested like Knowledge-based and Community-based techniques. Both techniques would probably allow to generate better and more interesting explanations to the users.

For offline evaluations, other metrics could have been used. There are other not addressed techniques less used than RMSE but they could give a very different result. Only implicit ratings were tested, a dataset with both implicit and explicit ratings would better depict a real system. User Studies would benefit from a larger user base. All user studies were based on profiles with watching data collected during one month. Testing the recommender with users with varying periods of user profiles would allow to check how the recommender works over time. Although the system is prepared to generate recommendations for every type of users, from the first time user to the recurring user, no tests were made to validate it. This system would also benefit from Online Evaluations. It would allow to have access to more users and faster algorithm iterations.

Finally, the user interface could be further developed to test different ways of presenting recommendations and explanations and to allow to provide feedback like telling the system it provided a wrong recommendation.

Appendix A

User Studies - Questions

User Profile

- Age
- Education Level

Television Usage

- Number of people that watch television at your home
  (Numeric Answer)
- How often do you watch TV at home?
  (Every Day, Every Week, Every Month, A couple of times per year, Never)
- How often do you watch TV with other people at home?
  (Every Day, Every Week, Every Month, A couple of times per year, Never)
- How long, in minutes, does it take on average to choose a programme to watch?
  (Numeric Answer)
- Which factors influence your decision of which programme to watch?
  (Free Text Input)
- What is your Cable TV operator?
  (Free Text Input)
- Does your Cable TV operator provide access to a Recommendation System?
  (Yes, No, I don't know)
- How often do you use your Cable TV recommendation system?
  (Always, Many times, Sometimes, Rarely, Never used)
- Are you satisfied when you use your Cable TV recommendation system?
  (Always, Many times, Sometimes, Rarely, Never used)

- How important are television recommendations systems for you?
  (Very Important, Important, Some Importance, Little Importance, No importance)
- How important are recommendation explanations for you?
  (Very Important, Important, Some Importance, Little Importance, No importance)
- In your opinion, where would you position a recommendation system?
  (Scale: 1 - Largely Conservative; 10 - Largely Serendipitous)

Evaluated System Feedback

- Evaluate the following criteria:
  (Scale: 1 - 10)
  - Recommendations Quality
  - Categories Order
  - Explanations Quality
- Evaluate your opinion on the following sentences:
  (Scale: 1 - Completely Disagree; 10 - Completely Agree)
  - I would use this recommendations system in the future
  - I am satisfied when I use this recommendations system
  - Explanations were important to decide whether the programme was of my interest or not
  - This recommendation system is efficient
  - This recommendation system is clear
  - With this recommendation system I'm more interested in watching programmes that I usually do not watch
- Comments
  (Free Text Input)

Explanation Types

- Evaluate the importance of the following category titles for recommended programmes:
  (Scale: 1 - Not Important; 10 - Very Important)
  - Popular Programmes
  - People like you also like
  - New episodes of your favorite programmes
  - Fans of [Favorite Programme] also like

  - Programmes with [Favorite Actor]
  - Programmes directed by [Favorite Director]
  - Programmes from the year [Favorite year]
  - Programmes broadcast in [Favorite TV Channel]
  - Programmes of [Favorite Category]
- Evaluate the importance of the following descriptions for recommended programmes:
  (Scale: 1 - Not Important; 10 - Very Important)
  - Because you watched a previous episode
  - Similar users also watched this programme
  - Similar users also watched programmes from the same category
  - Popular programme
  - Because you watched [Programme]
  - Because you like [Actor]
  - Because you like [Director]
  - General User Comments

User Comments

(Free Text Input)

Bibliography

[1] Jorge Abreu, Pedro Almeida, and Bruno Teles. Tv discovery & enjoy: A new approach to help users finding the right tv program to watch. In *Proceedings of the 2014 ACM International Conference on Interactive Experiences for TV and Online Video*, TVX '14, pages 63–70, New York, NY, USA, 2014. ACM.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[3] Kamal Ali and Wijnand van Stam. Tivo: Making show recommendations using a distributed collaborative filtering architecture. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 394–401, New York, NY, USA, 2004. ACM.

[4] Xavier Amatriain. Big & personal: Data and models behind netflix recommendations. In *Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, BigMine '13, pages 1–6, New York, NY, USA, 2013. ACM.

[5] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. A recommender system for an iptv service provider: a real large-scale production environment. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 299–331. Springer US, 2011.

[6] Patrick Baudisch and Lars Brueckner. Tv scout: Lowering the entry barrier to personalized tv program recommendation. In Paul De Bra, Peter Brusilovsky, and Ricardo Conejo, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 2347 of *Lecture Notes in Computer Science*, pages 58–68. Springer Berlin Heidelberg, 2002.

[7] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[8] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[9] Keng-Hao Chang, Jeffrey Hightower, and Branislav Kveton. Inferring identity using accelerometers in television remote controls. In *Proceedings of the 7th International Conference on Pervasive Computing*, Pervasive '09, pages 151–167, Berlin, Heidelberg, 2009. Springer-Verlag.

[10] Paul Cotter and Barry Smyth. Ptv: Intelligent personalised tv guides. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 957–964. AAAI Press, 2000.

[11] Duco Das and Herman ter Horst. Recommender systems for tv. 1998.

[12] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[13] Instituto Nacional de Estatística. Subscribers of subscription television service. `http://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_indicadores&indOcorrCod=0006870`, acessed in 2014.

[14] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, January 2004.

[15] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 107–144. Springer US, 2011.

[16] Digitalsmiths. *Q1 2014 Video Discovery Trends Report: Consumer Behavior Across Pay-TV, VOD, OTT, Connected Devices and Next-Gen Features*. Durham, NC, USA, 2014.

[17] Simon Dooms, Toon De Pessemier, and Luc Martens. Movietweetings: a movie rating dataset collected from twitter. In *Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013*, 2013.

[18] Michael Ehrmantraut, Theo Härder, Hartmut Wittig, and Ralf Steinmetz. The personal electronic program guide - towards the pre-selection of individual tv programs.

In *Proceedings of the Fifth International Conference on Information and Knowledge Management*, CIKM '96, pages 243–250, New York, NY, USA, 1996. ACM.

[19] Michael D. Ekstrand, Michael Ludwig, Joseph A. Konstan, and John T. Riedl. Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 133–140, New York, NY, USA, 2011. ACM.

[20] Simon Funk. Netflix update: Try this at home. `http://sifter.org/~simon/journal/20061211.html`, 2006.

[21] Jon Herlocker, JosephA. Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5(4):287–310, 2002.

[22] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 241–250, New York, NY, USA, 2000. ACM.

[23] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.

[24] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *In IEEE International Conference on Data Mining (ICDM 2008*, pages 263–272, 2008.

[25] Rong Jin, Luo Si, ChengXiang Zhai, and Jamie Callan. Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 309–316, New York, NY, USA, 2003. ACM.

[26] Thorsten Joachims and Filip Radlinski. Search engines that learn from implicit feedback. *Computer*, 40(8):34–40, August 2007.

[27] Christopher Krauss, Lars George, and Stefan Arbanowski. Tv predictor: Personalized program recommendations to be displayed on smarttvs. In *Proceedings of the 2Nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, BigMine '13, pages 63–70, New York, NY, USA, 2013. ACM.

[28] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. *CoRR*, abs/cs/0702144, 2007.

[29] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[30] Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1097–1101, New York, NY, USA, 2006. ACM.

[31] ZON Multimédia. *Impactos ZON na economia portuguesa 2007-2012*. Lisboa, Portugal, 2013.

[32] Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Giovanni Semeraro, Marco de Gemmis, Mauro Barbieri, Jan H. M. Korst, Verus Pronk, and Ramon Clout. Tv-show retrieval and classification. In Giambattista Amati, Claudio Carpineto, and Giovanni Semeraro, editors, *IIR*, volume 835 of *CEUR Workshop Proceedings*, pages 179–182. CEUR-WS.org, 2012.

[33] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, pages 249–256, New York, NY, USA, 1990. ACM.

[34] Douglas Oard and Jinmook Kim. Implicit feedback for recommender systems. In *in Proceedings of the AAAI Workshop on Recommender Systems*, pages 81–83, 1998.

[35] Pearl Pu and Li Chen. Trust building with explanation interfaces. In *Proceedings of the 11th International Conference on Intelligent User Interfaces*, IUI '06, pages 93–100, New York, NY, USA, 2006. ACM.

[36] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[37] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, March 1997.

[38] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer US, 2011.

[39] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000.

[40] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011.

[41] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[42] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, ICDEW '07, pages 801–810, Washington, DC, USA, 2007. IEEE Computer Society.

[43] Nava Tintarev and Judith Masthoff. Designing and evaluating explanations for recommender systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 479–510. Springer US, 2011.

[44] Jeroen Van Barneveld and Mark Van Setten. Designing usable interfaces for tv recommender systems. In *Personalized Digital Television*, volume 6 of *Human-Computer Interaction Series*, pages 259–285. Springer Netherlands, 2004.

[45] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM.